Esteban Alfaro Cortés

COMBINACIÓN DE CLASIFICADORES MEDIANTE EL MÉTODO BOOSTING. UNA APLICACIÓN A LA PREDICCIÓN DEL FRACASO EMPRESARIAL EN ESPAÑA

I.S.B.N. Ediciones de la UCLM
84-8427-433-0

Ediciones de la Universidad
de Castilla-La Mancha

Cuenca, 2006

UNIVERSIDAD DE CASTILLA-LA MANCHA
DEPARTAMENTO DE ECONOMÍA Y EMPRESA
FACULTAD DE CIENCIAS ECONÓMICAS Y EMPRESARIALES DE ALBACETE
ÁREA DE ESTADÍSTICA

COMBINACIÓN DE CLASIFICADORES MEDIANTE EL MÉTODO BOOSTING. UNA APLICACIÓN A LA PREDICCIÓN DEL FRACASO EMPRESARIAL EN ESPAÑA

Memoria presentada al departamento de Economía y
Empresa de la Universidad de Castilla-La Mancha
para la obtención del título de Doctor en
Ciencias Económicas y Empresariales

Presentada por:
Esteban Alfaro Cortés
Dirigida por:
Dr. D. Matías Gámez Martínez

Albacete, Septiembre de 2005.

Agradecimientos

Al Dr. D. Matías Gámez Martínez, director de esta Tesis Doctoral, por su valiosísima ayuda profesional y largas horas de dedicación durante la realización del presente trabajo, y que, además de director, también es un gran compañero y amigo.

Al Dr. D. José María Montero Lorenzo porque, a pesar de la distancia, siempre me ha aportado su experiencia, buenas ideas y ganas de trabajar.

A la Universidad de Castilla-La Mancha y, en particular, a la Facultad de Ciencias Económicas y Empresariales de Albacete, por las facilidades y ayuda recibidas.

A la empresa Informa, S.A. y, en especial, a Marta de la Fuente responsable de la cuenta de la Universidad de Castilla La Mancha, ya que sin su ayuda no habría sido posible la realización del presente trabajo.

A mi familia, en especial a mis hermanos y mis padres Esteban y María Pilar, por su comprensión y ayuda en los momentos necesarios y por enseñarme desde pequeño que *"todo está en los libros"*.

A mis compañeros de la Universidad de Castilla-La Mancha, en especial al Área de Estadística del Departamento de Economía y Empresa y a Noelia García Rubio, Gonzalo García Donato-Layrón y Montserrat Manzaneque Lizano por sus enseñanzas, aportaciones y gran apoyo.

A mis amigos y amigas, porque hacen que la vida sea más agradable.

A todos ellos, y muchos más que no he nombrado pero que me han ayudado a llegar hasta aquí, gracias.

ÍNDICE

CAPÍTULO 0. PLANTEAMIENTO Y OBJETIVOS 1

0.1. Planteamiento 1

0.2. Objetivos 4

PRIMERA PARTE

CAPÍTULO 1. INTRODUCCIÓN 11

1.1. Introducción 11

1.2. Definición de clasificación 12

1.2.1. ¿Por qué se construyen sistemas de clasificación? 13

1.2.2. Características deseables 14

1.2.3. Definiciones de las clases 15

1.2.4. Precisión 16

1.3. Ejemplos de clasificadores 17

1.3.1. Discriminante Lineal de Fisher 18

1.3.2. Árboles de decisión y métodos basados en reglas 19

1.3.3. K-vecinos más próximos 20

CAPÍTULO 2. CLASIFICACIÓN 23

2.1. Introducción 23

2.2. Estructura general de los problemas de clasificación 24

2.2.1. Probabilidad a priori y la regla por defecto 25

2.2.2. Separación de las clases 26

2.2.3. Los costes de una clasificación errónea 26

2.3. La regla de Bayes a posteriori 27

2.3.1. La regla de Bayes en estadística 30

2.4. Clasificación de los procedimientos de clasificación 31

2.4.1. Extensiones del discriminante lineal 31

2.4.2. Árboles de decisión y métodos basados en reglas 32

2.4.3. Estimaciones de la densidad 32

2.5. La selección de variables 32

2.5.1. Ventajas de la selección de variables 32

2.5.2. ¿Qué es la relevancia? 34

2.5.3. Selección de atributos como búsqueda heurística 39

2.5.4. Métodos incrustados de selección de atributos 40

2.5.5. Métodos de filtrado para la selección de atributos 41

2.5.6. Métodos envolventes para la selección de atributos 42

2.5.7. Métodos que ponderan los atributos 43

2.6. Estimación de la precisión de un sistema de clasificación 44

2.6.1. Tasa de error aparente 45

2.6.2. Estimación de la tasa de error real 46

2.6.3. Estimación por conjunto de prueba 47

2.6.4. Submuestreo aleatorio 48

2.6.5. Validación cruzada 49

2.6.6. Leaving one out 50

2.6.7. Bootstrapping 51

2.6.8. El error estándar 53

CAPÍTULO 3. MÉTODOS DE CLASIFICACIÓN INDIVIDUALES:

ANÁLISIS DISCRIMINANTE Y VECINO MÁS PRÓXIMO 55

3.1. Introducción 55

3.2. Discriminante Lineal 57

3.2.1. Obtención de las ecuaciones de las funciones discriminantes 60

3.2.2. Contrastes en el análisis discriminante 64

3.2.3. Interpretación de las funciones discriminantes y cálculo de la contribución de cada variable 67

3.2.4. Selección de variables 69

3.2.5. Clasificación de los individuos 72

3.3. Discriminante Cuadrático 76

3.3.1. La aplicación práctica del discriminante cuadrático 76

3.3.2. Regularización y estimaciones alisadas 78

3.4. Discriminante Logístico 79

3.4.1. La aplicación práctica del discriminante logístico 82

3.5. Método del vecino más próximo 84

3.5.1. La regla 1-NN 85

3.5.2. La regla *k*-NN 86

3.6. Cotas de error de las reglas 1-NN y *k*-NN 86

3.7. Clases de rechazo 87

3.8. Comentarios sobre las reglas *k*-NN 89

3.9. Reducción en las reglas del vecino más próximo 92

3.9.1. Regla del vecino más próximo condensado 92

3.9.2. Regla del vecino más próximo selectiva 95

3.9.3. Regla del vecino más próximo reducida 97

3.9.4. Regla del vecino más próximo editada 98

3.9.5. Todos los *k*-NN 99

3.9.6. Medida de similaridad entre variables 99

CAPÍTULO 4. MÉTODOS DE CLASIFICACIÓN INDIVIDUALES II:

REDES NEURONALES ARTIFICIALES Y ARBOLES DE CLASIFICACIÓN 103

4.1. Introducción 103

4.2. Motivación biológica de las redes neuronales artificiales 106

4.3. El perceptrón simple 107

4.3.1. La regla de entrenamiento del perceptrón 109

4.3.2. La regla delta o del gradiente descendente 112

4.4. Redes Multicapa 118

4.4.1. El algoritmo de retropropagación 121

4.4.2. Derivación de la regla de retropropagación 122

4.4.3. Inclusión del impulso 125

4.5. La aplicación práctica del algoritmo de retropropagación 126

4.6. Árboles de clasificación 130

4.7. Construcción del árbol de clasificación 133

4.8. La regla de corte o división 134

4.8.1. Criterios de corte 135

4.8.2. Bondad de una partición 139

4.8.3. Impureza de un árbol 140

4.9. Criterio de parada 141

4.10. Sobreajuste en los árboles de clasificación 143

4.11. Técnicas de podado 145

4.11.1. Podado por mínimo coste-complejidad 146

4.11.2. Podado de reglas 149

4.12. Valores omitidos 152

SEGUNDA PARTE

CAPÍTULO 5. PROBLEMÁTICA DE LOS CLASIFICADORES INDIVIDUALES 155

5.1. Introducción 155

5.2. Descomposición del error. Estudio del sesgo y de la varianza 158

5.2.1. Estimaciones bootstrap del sesgo y de la varianza 163

5.3. Estudio de la inestabilidad de los clasificadores 166

5.4. ¿Por qué utilizar combinaciones de clasificadores? 171

5.5. La perspectiva bayesiana sobre la agregación de clasificadores 175

CAPÍTULO 6. MÉTODOS DE COMBINACIÓN DE CLASIFICADORES 177

6.1. Introducción 177

6.2. Clasificación de los métodos de combinación 180

6.2.1. Métodos no generadores 181

6.2.2. Métodos generadores 182

6.3. Bagging 184

6.4. Boosting 188

6.4.1. Error de entrenamiento en Adaboost 192

6.4.2. La teoría del margen y Adaboost 193

6.4.3. Comparación de Bagging y Boosting 196

6.5. Bosques aleatorios 196

6.5.1. Estimaciones out-of-bag 201

CAPÍTULO 7. EL MÉTODO BOOSTING Y SUS VARIACIONES 203

7.1. Introducción 203

7.2. Los orígenes del Boosting 206

7.3. Relación del método Boosting con los modelos aditivos generalizados 211

7.4. Versiones de Adaboost 218

7.5. Generalización del Boosting al caso de q clases 223

7.6. El tamaño de los árboles en Boosting 228

TERCERA PARTE

CAPÍTULO 8. PREDICCIÓN DEL FRACASO EMPRESARIAL 235

8.1. Introducción 235

8.2. Estado actual de la previsión del fracaso empresarial 237

8.2.1. Los trabajos pioneros 240

8.2.2. Superación del Análisis Discriminante Lineal 241

8.2.3. Los ratios financieros utilizados 246

8.3. Criticas y ventajas de los modelos de predicción del fracaso empresarial 249

8.4. Análisis descriptivo de las estadísticas sobre suspensiones de pagos, quiebras, fusiones y disoluciones 251

8.4.1. Suspensiones de pagos 253

8.4.2. Quiebras 258

8.4.3. Sociedades mercantiles disueltas y fusionadas 263

8.4.4. Evolución comparada de los distintos tipos de fracaso 265

CAPÍTULO 9. PREDICCIÓN DEL FRACASO EMPRESARIAL MEDIANTE EL MÉTODO BOOSTING 269

9.1. Introducción 269

9.2. Algunas consideraciones teóricas sobre el tratamiento de la información 271

9.2.1. Introducción 271

9.2.2. Obtención del conjunto final de variables de entrada 272

9.2.3. Análisis previo de los datos 273

9.2.3.1. - Detección de valores atípicos 273

9.2.3.2. - Detección de existencia de ruido 274

9.2.3.3. - Tratamiento de datos ausentes 276

9.2.4. Preprocesamiento de los datos 277

9.3. Recogida y tratamiento de la información 279

9.3.1. Fuentes de información 279

9.3.2. Tratamiento de la información 281

9.4. Aplicación del método boosting a la previsión del fracaso empresarial 308

9.4.1. Predicción del fracaso empresarial con árboles de clasificación. El caso de dos clases 309

9.4.2. Predicción del fracaso empresarial mediante Boosting. El caso de dos clases 316

9.4.3. Predicción del fracaso empresarial con árboles de clasificación. El caso de tres clases 322

9.4.4. Predicción del fracaso empresarial mediante Boosting. El caso de tres clases 329

9.4.5. Comparación de boosting con otros métodos 337

9.4.6. Predicción del fracaso empresarial ampliando el horizonte temporal 342

9.4.7. Resumen de la aplicación del método boosting a la predicción del fracaso empresarial 347

CAPÍTULO 10. CONCLUSIONES Y LÍNEAS DE INVESTIGACIÓN FUTURA 349

10.1. Conclusiones 349

10.2. Líneas de investigación futura 362

BIBLIOGRAFÍA 367

APÉNDICE A: ALGORITMOS DE CLASIFICACIÓN 405

A.1. Función Adaboost.m1 405

A.2. Función boosting.cv 408

A.3. Función predict.boosting 409

A.4. Función bagging 410

A.5. Función bagging.cv 412

A.6. Función predict.bagging 413

A.7. Función margins 414

A.8. Ayuda de la futura librería de R 415

APÉNDICE B: SALIDA DE LA FUNCIÓN ADABOOST.M1 427

B.1. Salida de la función Adaboost.M1 427

CAPÍTULO 0

PLANTEAMIENTO Y OBJETIVOS

0.1. PLANTEAMIENTO

El doctorando presenta el trabajo de investigación titulado "*Combinación de clasificadores mediante el método boosting. Una aplicación a la predicción del fracaso empresarial en España*", para la obtención del grado de Doctor dentro del Área de Economía Aplicada de la Facultad de Ciencias Económicas y Empresariales de Albacete, perteneciente a la Universidad de Castilla-La Mancha. Dicho estudio se circunscribe al ámbito del conjunto del territorio nacional debido a que de esta manera su aplicación puede alcanzar un grado más generalizado. Además, la Universidad a la que pertenece mantiene una relación estrecha con el entorno empresarial y social. Por consiguiente, parece adecuado que contribuya mediante la presente investigación a la

provisión de resultados económicos que puedan ser de utilidad al sector empresarial y, por ende, a la sociedad.

Delimitado el ámbito territorial de la investigación se ha de señalar, en segundo lugar, que se ha ubicado en el sector empresarial debido a su importancia, por todos conocida, en la actividad económica en general y, en la de este país en particular. Las empresas son sin duda el motor de nuestra economía ya que transforman el ahorro en inversión y, con ello contribuyen a la generación de valor añadido. Además, tienen un papel especialmente importante en el empleo ya que según la Encuesta de Población Activa (EPA) en España, del total de ocupados, prácticamente un 66% son asalariados del sector privado. En concreto de 18.492.700 ocupados, 12.145.500 están en esa situación profesional según la EPA del primer trimestre de 2005.

Por otro lado, las consecuencias del fracaso empresarial no se limitan a las personas, empresas u organizaciones que han establecido de forma directa una relación con la empresa fallida. A menudo estas consecuencias se extienden al entorno empresarial y, por ende, al conjunto de la economía y al entorno social de un país o de una región. Por ejemplo, los países o regiones menos desarrollados son, generalmente, muy sensibles a los fracasos de sus empresas, especialmente, si éste afecta a alguna de las empresas con un importante impacto en la economía regional o nacional. Además, teniendo en cuenta la globalización del entorno económico, incluso puede llegar a tener consecuencias globales, como ocurrió en la reciente crisis del sudeste asiático.

Todo esto pone de manifiesto la importancia de desarrollar e implementar procedimientos eficientes para la predicción del fracaso empresarial. Estos procedimientos son útiles para instituciones financieras, personas e instituciones inversoras, así como, para las propias empresas e incluso para las autoridades públicas (p.ej. gobiernos y bancos centrales).

El pronóstico del fracaso empresarial es un campo de investigación que se ha extendido a lo largo de casi cuatro décadas y que, todavía hoy, sigue presentando un

elevado interés en la comunidad científica. En este tiempo ha habido numerosas mejoras metodológicas, que no siempre han conseguido una mejora importante de los resultados conseguidos. En la actualidad, este campo ha tomado un nuevo impulso debido a la mayor disponibilidad de información y a la utilización de modelos alternativos que provienen del aprendizaje automático. Además, se están probando variables cualitativas que permitan, junto con las variables contables numéricas tradicionales, incrementar la precisión de los modelos para la predicción, así como, el conocimiento de los factores que determinan la futura continuidad o no de la empresa.

En tercer lugar, la metodología propuesta en el presente trabajo, a saber, el método boosting, es muy novedosa. Este método se basa en la idea de que combinando varios clasificadores básicos que consigan un error sólo ligeramente mejor que el azar es posible potenciar la precisión de estos clasificadores y conseguir que el clasificador combinado obtenga una precisión muy elevada.

El método boosting constituye una herramienta nueva y potente para el tratamiento de la información, que complementa a las técnicas estadísticas tradicionales. Este método es capaz de resolver problemas en los que las técnicas tradicionales no son adecuadas. Además, en los casos donde se pueden aplicar ambos, boosting mejora los resultados proporcionados por ellas.

El uso de este método es especialmente indicado para abordar problemas que, por sus características, exigen un nivel de error muy reducido o, lo que es lo mismo, un nivel de acierto muy elevado. Además, no requiere supuestos restrictivos, como por ejemplo la normalidad de las variables para el análisis discriminante lineal.

Las circunstancias antes señaladas aparecen, sin duda alguna, en el campo de aplicación de este trabajo, donde:

- Se trabaja con ratios financieros que, como se comprobará en la aplicación, no suelen distribuirse según una normal.

< El nivel de error debe reducirse al mínimo. Especialmente, lo que se conoce como error de tipo I, es decir, el error que se comete al clasificar una empresa que va a fracasar como empresa sana. Esto se debe a que las consecuencias de este error son, en la mayoría de los casos, mucho más graves que las de pronosticar el fracaso de una empresa que en realidad seguirá activa.

Una vez puesta de manifiesto la importancia del problema de la predicción del fracaso empresarial en España, se detallan a continuación los objetivos que se persiguen al comenzar este trabajo.

0.2. OBJETIVOS

Considerando plenamente justificado el estudio que se plantea en este trabajo por la importancia socio-económica del sector empresarial, por el interés de seguir perfeccionando el pronóstico del fracaso empresarial, y por lo novedoso de la técnica que se propone, se van a recoger de forma resumida los objetivos, tanto metodológicos como empíricos, que se pretenden alcanzar:

1. En primer lugar, se quiere proporcionar una **visión general del abanico de métodos de clasificación individuales más utilizados**. Dentro de estos métodos se hará distinción entre métodos paramétricos y no paramétricos. De cada uno de ellos se destacarán sus principales características, así como las ventajas e inconvenientes que presentan.

2. Se desea **demostrar la utilidad y conveniencia de trabajar con combinaciones de clasificadores** para superar algunas limitaciones que presentan los métodos de clasificación individuales. Además, debido a la gran variedad de métodos de combinación que existen se establecerá una clasificación que ayude a comprender mejor su naturaleza y las diferencias entre ellos.

3. En tercer lugar, se pretende **analizar con detenimiento el método boosting**, sus antecedentes y estado actual. Este método es la técnica principal de este trabajo, por lo que se estudian las versiones más relevantes del mismo y algunas consideraciones teóricas respecto a este método.

4. **Estudio de la literatura previa en predicción del fracaso empresarial**, analizando los orígenes, su evolución y el estado actual del tema. Además, deben seleccionarse los ratios financieros que han resultado de mayor utilidad para el pronóstico del fracaso. También es conveniente realizar una descripción de la evolución que han seguido en España las empresas fracasadas.

5. Consideración de un **concepto de fracaso empresarial más amplio** del habitual. Normalmente, se consideran únicamente las empresas que han entrado en un proceso concursal de suspensión de pagos o quiebra. Obviamente el concepto de fallo de una empresa es más extenso. En este trabajo se incluyen también como empresas fracasadas a aquellas empresas que han sido disueltas y aquellas otras que han sido absorbidas por otras empresas. De esta forma no se considera sólo el fracaso legal sino también el fracaso como cese de la actividad.

6. **Discriminación entre tres clases, activas, fracaso legal y fracaso como cese**. Se pretende ir más allá del planteamiento dicotómico del problema de predicción entre empresas fracasadas y sanas (activas). Para ello se trabajará con tres clases, activas, *fracaso1*(fracaso como cese) y *fracaso2* (fracaso legal).

7. **Utilización de información cualitativa adicional** a la información cuantitativa tradicional. La mayoría de los trabajos se limitan a trabajar con ratios financiero-contables. Sin poner en duda la incuestionable utilidad de esta información, dada la complejidad del problema, parece conveniente la inclusión de información cualitativa relativa al tamaño de la empresa, la actividad a la que se dedica y, tal vez, la forma jurídica que posee.

8. **Contrastación empírica de la metodología propuesta** mediante su aplicación práctica con datos de empresas españolas incluidas dentro de las tres clases que se llegarán a distinguir, aunque al principio los dos tipos de fracaso serán considerados conjuntamente. Además, para comprobar la bondad o no de sus resultados se llevará a cabo una comparación entre el método boosting y otros métodos, tanto individuales como combinados. Por último, se estudia también los resultados de la predicción al retroceder más respecto del momento en que se produce el fracaso.

Para conseguir los objetivos que se acaban de plantear el trabajo se ha estructurado en tres partes bien diferenciadas que comprenden en total diez capítulos, a parte de este capítulo cero que justifica el interés de la cuestión y plantea los objetivos de la investigación.

La primera parte comprende del capítulo I al capítulo IV. Los dos primeros tienen por objetivo exponer algunos aspectos generales de los problemas de clasificación, tales como la estructura general, la regla óptima de clasificación a la que todos los demás sistemas intentan imitar, la selección de los atributos o características que describen a cada ejemplo y, por último, las diferentes formas de medir la precisión. Los capítulos III y IV explican algunos de los métodos de clasificación individuales más utilizados en la actualidad, divididos de acuerdo a los siguientes grupos: técnicas estadísticas tradicionales o clásicas, el método del vecino más próximo, las redes neuronales y, finalmente, los árboles de clasificación.

En la segunda parte (capítulos V al VII) se analizan algunos aspectos relacionados con el comportamiento y las propiedades de estos clasificadores individuales. En concreto, se plantean las dificultades que pueden surgir debido al uso de los clasificadores individuales, como son la precisión y la estabilidad de los mismos. A continuación, en el capítulo VI, se aborda el estudio de la combinación de clasificadores prestando especial atención al método boosting. Además, se recoge una taxonomía de los métodos de combinación y se introducen también el método bagging y el bosque aleatorio. Por último, se estudian los primeros algoritmos que han dado lugar al

desarrollo posterior del método boosting. También se exponen algunas de las modificaciones que se han propuesto al algoritmo Adaboost, incluyendo las que sirven para afrontar la existencia de más de dos clases y, para acabar, se analiza cuál debe ser el tamaño adecuado de los árboles utilizados en la combinación.

En la tercera parte, que incluye los tres últimos capítulos, se proporcionará una visión general de la predicción del fallo empresarial, sus antecedentes y estado actual. Además, se elaborará un listado con los ratios financieros que han resultado de mayor utilidad para el pronóstico del fracaso. También se lleva a cabo una descripción de la evolución que han seguido en España las empresas fracasadas. El capítulo IX se centra en la aplicación práctica. Después de recoger brevemente algunas consideraciones teóricas sobre el tratamiento de la información, se realiza un análisis exploratorio de los datos. Además de catorce ratios financieros, se utilizan otras tres variables menos habituales que intentan recoger el tamaño de la empresa, la actividad a la que se dedica y la forma jurídica que presenta. Se coteja el método boosting con los árboles de clasificación, tanto para el caso dicotómico, como cuando se distingue entre tres clases. A continuación se realiza una comparación, algo menos detallada, con otros cinco métodos de clasificación. Posteriormente, se examina la capacidad de los modelos establecidos anteriormente para predecir el fracaso empresarial cuando aumenta la distancia temporal al período en que se hace efectivo el fallo. Finalmente, en el capítulo X se expondrán las principales conclusiones derivadas de la aplicación práctica y del conjunto de esta investigación.

PRIMERA PARTE

CAPÍTULO 1

INTRODUCCIÓN

1.1. INTRODUCCIÓN

El problema de la clasificación está presente en un amplio rango de la actividad humana. En su forma más general el término puede cubrir cualquier contexto en el que se toma una decisión o se realiza una predicción en base a la información disponible en ese momento, y un procedimiento de clasificación es, entonces, un método formal para repetir esos razonamientos en situaciones nuevas.

Este trabajo se centra en una interpretación algo más concreta. El problema consiste en construir un procedimiento que se aplicará sobre una secuencia conjunta de casos, en los que cada nuevo caso debe ser asignado a una, de entre un conjunto de clases predefinidas, basándose en atributos o características observadas.

La construcción de un sistema de clasificación a partir de un conjunto de datos para el cual se conocen las verdaderas clases, ha sido denominada de diferentes formas: reconocimiento de patrones, análisis discriminante o aprendizaje supervisado. Ésta última denominación, se realiza para distinguirlo del aprendizaje no supervisado o agrupamiento en el que las clases se infieren a partir de los datos.

El problema de clasificación está presente en contextos tan distintos como, por ejemplo, el procedimiento mecánico para enviar las cartas en base a la lectura automática de los códigos postales, la toma de decisiones respecto a las solicitudes de crédito de los individuos en base a información financiera y otra información personal y el diagnóstico preliminar de la enfermedad de un paciente para seleccionar un tratamiento inmediato mientras se espera a los resultados definitivos de las pruebas.

1.2. DEFINICIÓN DE CLASIFICACIÓN

Cuando se habla de clasificación puede tener dos significados distintos, se puede tener un conjunto de observaciones con el objetivo de establecer la existencia de clases o grupos en los datos. O se puede saber que existen determinadas clases, y que el objetivo sea establecer una regla por la que se llegue a clasificar una nueva observación dentro de una de las clases existentes. El primer tipo se conoce como *aprendizaje no supervisado* (o clustering), el segundo como aprendizaje supervisado. Este trabajo se centra en el segundo, *aprendizaje supervisado*. En estadística tradicionalmente se ha utilizado para este propósito el análisis discriminante pero en los últimos tiempos se han desarrollado nuevas técnicas, en parte gracias al desarrollo experimentado en la capacidad de los soportes informáticos.

Los métodos de clasificación han sido utilizados en muchas disciplinas biología (genética), medicina (diagnóstico de enfermedades), astronomía, ingeniería, control y robótica. En economía son muchas las posibles aplicaciones de los métodos de clasificación, que intentan comprender la pertenencia a un grupo, por ejemplo, clasificación de los individuos en clientes y no clientes para una empresa, separación

de las empresas con riesgo de fracasar de aquellas que no lo tienen, determinación de las claves que conducen a un nuevo producto al éxito o al fracaso. Establecer la categoría de riesgo asociada a una solicitud de crédito, tanto en el caso de créditos personales como de empresas.

Los inversores deben decidir comprar o vender las acciones en base a la información de la propia empresa y de la economía en general. Las autoridades fiscales deben decidir cuando hay que realizar una inspección, basándose en información financiera y fiscal, tanto en el caso de empresas como de personas. Las autoridades financieras (Banco de España, Comisión Nacional del Mercado de Valores, etc.) tienen que decidir sobre la intervención o no de una institución financiera. En el mercado de la vivienda, se puede utilizar un sistema de clasificación para determinar, de forma objetiva, el grado de calidad de una vivienda. En el marco de la economía regional es interesante establecer los parámetros que determinan la pertenencia de una región a un grupo ya que en función de esto se determinan las ayudas a recibir. En control de calidad se distingue entre productos defectuosos y no defectuosos.

La existencia de datos correctamente clasificados presupone que alguien (experto o supervisor) es capaz de clasificar sin error, por lo que alguien se podría preguntar, ¿por qué reemplazar esta clasificación exacta por una aproximación?

1.2.1. ¿Por qué se construyen sistemas de clasificación?

Entre las razones para hacer un procedimiento de clasificación, se pueden destacar:

1. Estos procedimientos de clasificación pueden ser más rápidos, por ejemplo las máquinas que leen los códigos postales, pueden clasificar la mayoría de las cartas dejando a los humanos sólo los casos difíciles.

2. Las decisiones deben tomarse en base a criterios objetivos iguales para todos los casos lo que, en el caso de los humanos, es difícil de conseguir pues pueden

verse afectados por factores externos por ejemplo de tipo psicológico, lo que originaría sesgos en las decisiones. Además en general se requieren varios expertos que puedan tomar la decisión lo que aumenta el grado de subjetividad. Por ejemplo en el caso de la concesión de créditos puede decidir en función de información irrelevante y perder un buen cliente

3. Algunos de los métodos de clasificación permiten explicar la pertenencia a una determinada clase en base a un conjunto de variables o atributos que describen a cada observación.

4. Además, disponer constantemente de expertos que tomen las decisiones puede ser mucho más costoso que desarrollar un sistema eficaz de clasificación a partir de la experiencia acumulada que pueda ser aplicado posteriormente por cualquier persona, no necesariamente experta en el tema pero que, siguiendo las pautas marcadas por el clasificador construido, pueda tomar las decisiones necesarias a partir de la información conveniente.

1.2.2. Características deseables

Hay algunas cuestiones en relación al posible clasificador que se tienen que considerar. En primer lugar, es deseable que tenga las siguientes propiedades:

Precisión: Es la consistencia de la regla, normalmente se representa por la proporción de clasificaciones correctas, aunque podría ser que algunos errores fuesen más importantes que otros y en este caso interesaría controlar el ratio de error para las clases más relevantes. En este trabajo, en principio, se consideran todas las clases de igual relevancia.

Velocidad: En determinadas circunstancias, la rapidez del clasificador es un tema importante. Un clasificador con un 90% de precisión, puede ser preferible a otro con una precisión del 95% si es mucho más rápido. En algunos casos las redes

neuronales consiguen clasificadores hasta 100 veces más rápidos. Estas consideraciones son especialmente importantes en casos como la detección automática de productos defectuosos en una cadena de producción. Aunque gracias a la potencia de las máquinas actuales esta característica no es tan decisiva.

Comprensibilidad: Si el procedimiento de clasificación debe ser aplicado por un humano, debe ser fácilmente entendible para evitar errores en dicha aplicación. Además, es importante también que el usuario crea en el sistema.

Tiempo para aprender: Sobre todo en un ambiente que cambia con rapidez, puede ser necesario aprender una regla de clasificación rápidamente, o reajustar la regla existente en tiempo real. “Rápidamente” puede implicar también que haya que utilizar un pequeño número de observaciones para establecer la regla.

Por una parte, si se considera la sencilla regla del vecino más próximo, donde en el conjunto de entrenamiento se busca el ejemplo previo más cercano (en un sentido definido), cuya clase se asume para el nuevo caso. Este método es muy rápido de aprender, pero muy lento de aplicar si se utilizan todos los datos. Por otro lado, hay casos donde es muy útil tener un método rápido, aunque sea poco preciso, para una rápida inspección de los datos o para una rápida comparación con los resultados de otro sistema. Por ejemplo, un empleado de banco debe saber la regla de no conceder, en general, créditos a aquellas personas que no tengan cuenta en el banco, por tanto, si un nuevo sistema automático concede reiteradamente créditos a personas que no tengan cuenta en el banco este empleado querrá asegurarse de que el nuevo sistema funciona correctamente.

1.2.3. Definiciones de las clases

Otro aspecto importante es la naturaleza de las clases y la forma en que son definidas. Se pueden distinguir tres casos:

1. Las clases se corresponden con etiquetas o nombres de poblaciones distintas, y la pertenencia a las poblaciones no es cuestionable. Por ejemplo, perros y gatos son dos clases o poblaciones muy diferentes, y se conoce con exactitud cuando un animal es un gato o un perro. Los miembros de una clase los determina una autoridad independiente (el supervisor o experto). La pertenencia a una clase no depende de ninguna variable o atributo.

2. Las clases resultan de un problema de predicción. Aquí la clase es esencialmente una salida que se debe predecir, conociendo los atributos. En estadística, la clase es una variable aleatoria. Un ejemplo típico es la predicción de los tipos de interés, donde normalmente se plantea si van a subir (clase =1) o no (clase =0).

3. Las clases están predefinidas mediante una partición del espacio muestral, por ejemplo los propios atributos. Se puede decir que la clase es una función de los atributos. Por ejemplo, un producto manufacturado puede ser defectuoso si algún atributo está fuera de los límites preestablecidos y no defectuoso en otro caso. Hay una regla que clasifica los datos a partir de los atributos, el problema es crear una regla que imite a la verdadera regla lo mejor posible. Muchos casos de concesión de crédito son de este tipo.

En la práctica los conjuntos de datos suelen ser mezclas de estos tipos, o bien algo intermedio.

1.2.4. Precisión

Al hablar de precisión, se debe tener en cuenta si se mide sobre el conjunto de entrenamiento o sobre nuevas observaciones (el conjunto de prueba) porque a menudo se obtienen resultados diferentes. De hecho esto no es raro, sobre todo en el aprendizaje automático algunas aplicaciones consiguen ajustarse perfectamente al conjunto de entrenamiento pero el comportamiento sobre el conjunto de test es decepcionante.

Normalmente, en la práctica lo que importa es la precisión sobre observaciones nuevas, donde la verdadera clasificación es desconocida.

El método generalmente aceptado para estimar la precisión consiste en utilizar el conjunto de observaciones disponible, suponiendo que se conoce la verdadera clase de todas ellas, de la siguiente forma. Primero se utiliza una parte sustancial de las observaciones para enseñar al sistema, esta parte se conoce como conjunto de entrenamiento[1]. La regla que se obtiene se aplica en las observaciones restantes (el conjunto de prueba o test), y los resultados se comparan con el verdadero valor de las clases. La proporción de aciertos en el conjunto de test es una estimación insesgada de la precisión de la regla gracias a que el conjunto de entrenamiento se obtiene mediante muestreo aleatorio sobre el conjunto de observaciones del que se partía. En el apartado 2.6 se trata más profundamente la estimación de la precisión, considerando otros métodos como la validación cruzada, leaving one out o bootstrapping.

1.3. EJEMPLOS DE CLASIFICADORES

Para ilustrar algunos tipos básicos de clasificadores, se utiliza el famoso conjunto de datos de los lirios, cuya descripción completa viene dada en Kendall y Stuart (1983) y que fue utilizado por Fisher en 1936 para establecer el análisis discriminante. Considera tres variedades de Lirios: Setosa, Versicolor, y Virgínica. Las variables clasificadoras son la longitud y anchura de cada pétalo y sépalo que se midieron en 50 flores de cada variedad. El problema original consistía en clasificar un nuevo lirio dentro de uno de esos tres tipos basándose en los cuatro atributos, longitud y amplitud del pétalo y el sépalo. Para simplificar este ejemplo, se buscará una regla de clasificación para distinguir las variedades solamente basándose en dos medidas, longitud y amplitud del pétalo, cuyo poder de discriminación es el mayor. Se dispone de 50 pares de medidas de cada variedad a partir de las cuales aprender la regla de clasificación.

[1] El conjunto de entrenamiento se conoce en inglés como training set.

1.3.1. Discriminante Lineal de Fisher

Es uno de los procedimientos de clasificación más antiguos y es el que mayoritariamente se utiliza en los paquetes estadísticos. La idea es dividir el espacio muestral mediante una serie de lineas en el espacio bidimensional, planos en el caso de tres dimensiones y, en general, hiperplanos para muchas dimensiones. La línea que divide dos clases se dibuja de tal forma que corte por la mitad la línea que une los centros de esas clases, la dirección de la línea se determina por la forma de los grupos de puntos. Por ejemplo, para diferenciar entre versicolor y virgínica, se aplica la siguiente regla:

- Si el ancho del pétalo < 3,272 - 0,3254*longitud pétalo, entonces versicolor
- Si el ancho del pétalo > 3,272 - 0,3254*longitud pétalo, entonces virgínica

El discriminante lineal de Fisher aplicado a los datos de los lirios se muestra en la figura 1.1. Seis de las observaciones se clasifican erróneamente.

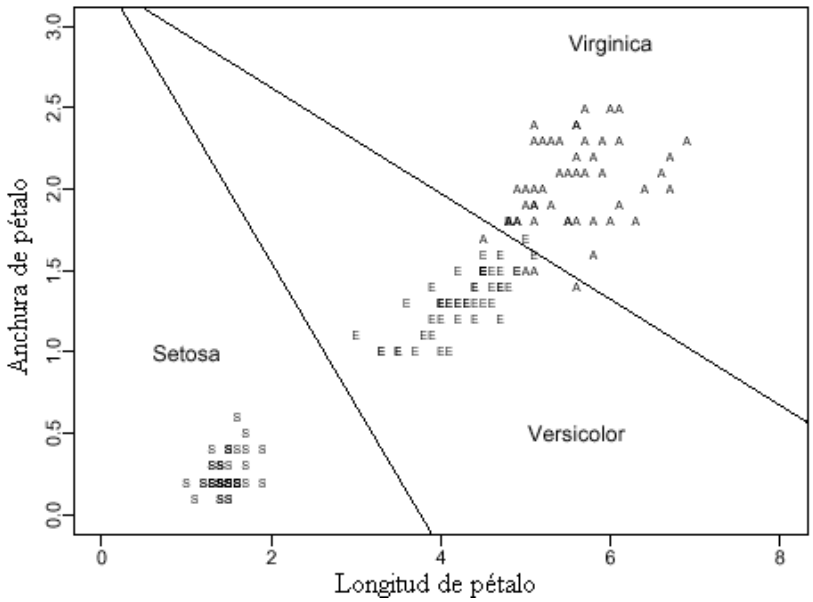


Figura 1.1 Clasificación mediante discriminante lineal. Conjunto de lirios

1.3.2. Árboles de decisión y métodos basados en reglas

Una clase de los procedimientos de clasificación se basa en particiones recursivas del espacio muestral. El espacio se divide en cajas (boxes), y en cada etapa del procedimiento se examina cada caja para ver si se puede separar en dos cajas, la división normalmente es paralela en todas las cajas. Un ejemplo para los datos de los lirios es el que sigue,

- Si la longitud del pétalo < 2,65 , entonces setosa
- Si la longitud del pétalo >4,95 , entonces virgínica
- Si 2,65< longitud del pétalo < 4,95 , entonces:
  - Si ancho del pétalo< 1,65 entonces versicolor
  - Si ancho del pétalo> 1,65 entonces virgínica

La partición resultante se muestra en figura 1.2. Nótese que esta regla de clasificación tiene tres errores.

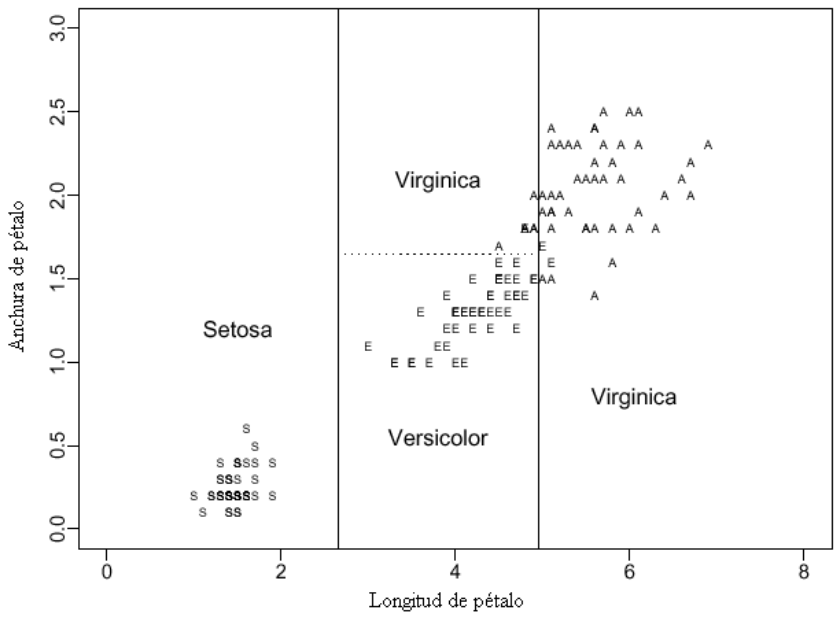


Figura 1.2 Clasificación por árboles de decisión. Conjunto de lirios

Weiss y Kapouleas (1989) dieron una regla de clasificación alternativa para los datos de los lirios que está muy directamente relacionada con la figura 1.2. Su regla puede obtenerse de la figura 1.2 continuando la línea de puntos hacia la izquierda y puede expresarse de esta forma:

- Si la longitud del pétalo < 2,65 , entonces setosa
- Si la longitud del pétalo >4,95 o anchura del pétalo>1,65, entonces virgínica
- En otro caso versicolor

Nótese, que esta regla aunque equivalente a la regla de la figura 1.2, se comporta de forma más precisa, y esta formulación puede ser preferible por esta razón. Obsérvese también que la regla es ambigua si la longitud del pétalo es inferior a 2,65 y el ancho de pétalo 1,65. Las reglas citadas pueden dejar de ser ambiguas, aplicándolas en el orden dado, y entonces serán sólo una reafirmación del árbol de decisión anterior. La regla aquí expuesta es un caso de los métodos de clasificación basados en reglas: estos métodos tienen vínculos muy cercanos con los árboles de decisión.

1.3.3. K-vecinos más próximos

Como en el caso anterior se muestra esta técnica sobre los datos de los lirios. Supóngase que hay que clasificar un nuevo caso de lirio. La idea es que es más probable que esté cerca de observaciones de su misma población. Se puede fijar la atención , por ejemplo, en las k (5) observaciones más cercanas de todas las almacenadas previamente y se clasifica la nueva observación de acuerdo con la clase mayoritaria entre sus vecinos. En la figura 1.3 la observación nueva está marcada con una +, y las 5 observaciones más cercanas rodeadas por el círculo centrado en la +. La forma aparentemente elíptica es consecuencia de las diferencias entre escala horizontal y vertical, pero encontrar una escala apropiada para las observaciones es una dificultad importante de este método.

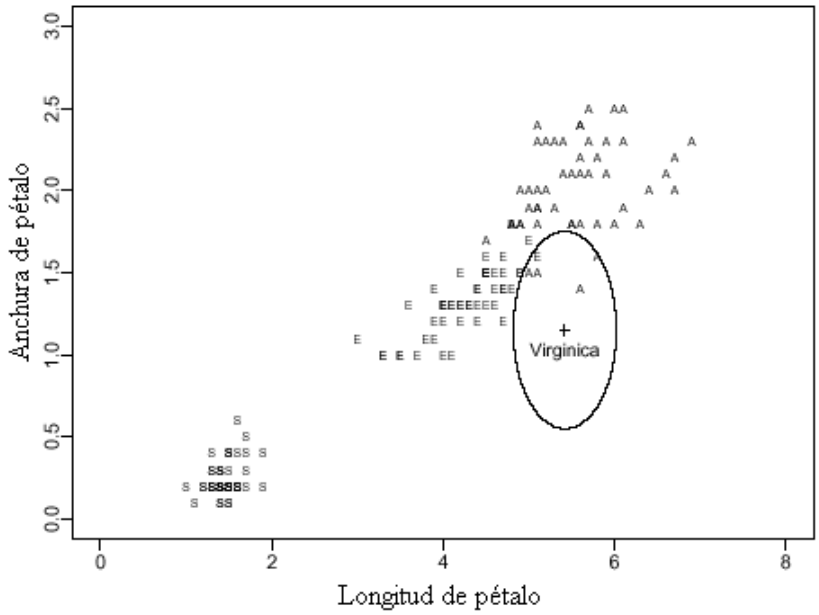


Figura 1.3 Clasificación mediante el vecino más próximo de orden 5. Conjunto de lirios

Esto se muestra en la figura 1.3, donde una observación centrada en + sería clasificada como virgínica porque tiene 4 observaciones de la clase virgínica entre sus 5 vecinos más próximos.

CAPÍTULO 2

CLASIFICACIÓN

2.1. INTRODUCCIÓN

En este capítulo se van a abordar algunos aspectos de carácter general relacionados con la clasificación, que en ocasiones son obviados o simplemente olvidados, pero que, no por ello dejan de tener un papel importante a la hora de entender las similitudes y diferencias entre los distintos procedimientos de clasificación que se verán a continuación.

En primer lugar, se expondrá el nexo común subyacente a todos los procedimientos de clasificación, ya que los problemas a los que se aplican presentan una estructura general común. En segundo lugar, se comentará la regla a posteriori de Bayes que es, la regla óptima de clasificación a la que intentan imitar.

Posteriormente, se establecerá una taxonomía de los distintos métodos de clasificación, intentando resaltar los puntos en común que existen entre muchos de ellos. En cuarto lugar, se plantearán las ventajas que presenta la selección de los atributos o características que describen cada ejemplo. Para ello, se seleccionan aquellas características con mayor poder discriminatorio, o lo que es lo mismo, las más relevantes. Ahora bien, para poder realizar esto se debe aclarar primero, qué es lo que se entiende por relevancia, por lo que se comentarán las definiciones que se han considerado más interesantes, de entre las que se han dado en la literatura de clasificadores.

Para acabar con este apartado se comentarán los distintos métodos que se pueden utilizar para llevar a cabo esta selección, pudiendo considerar dos grandes grupos, por un lado, aquellos que consideran la relevancia como un concepto relativo y atribuyen pesos a los atributos en función de la misma y, por otro, los métodos mas radicales que eliminan aquellos atributos que consideran irrelevantes o lo que es lo mismo sólo utilizan los relevantes.

Para finalizar el tema se centrará al análisis en cómo evaluar el comportamiento de un clasificador. Para ello se verán los distintos métodos que permiten estimar, con mayor o menor acierto, la precisión del clasificador ante nuevas observaciones. La estimación de la precisión del clasificador se realiza utilizando ciertas técnicas estadísticas de muestreo (remuestreo aleatorio, bootstrap, etc.), que se comentarán en ese apartado.

2.2. ESTRUCTURA GENERAL DE LOS PROBLEMAS DE CLASIFICACIÓN

Hay tres componentes esenciales en un problema de clasificación:

1. La frecuencia relativa de las clases en la población de interés, expresada formalmente como distribución de probabilidad a priori.

2. Un criterio implícito o explícito para separar las clases, se debe pensar en una relación subyacente de entrada o salida que utilice los atributos observados para distinguir un individuo aleatorio de una clase.

3. El coste asociado con clasificar erróneamente a un individuo.

Algunas técnicas confunden implícitamente estos componentes y, por ejemplo, producen una regla de clasificación que está condicionada por una distribución a priori particular y que no puede adaptarse fácilmente a un cambio en la frecuencia de la clase. Sin embargo, en teoría, cada uno de estos componentes deben ser estudiados individualmente y después combinar formalmente los resultados en una regla de clasificación. A continuación se describe este desarrollo.

2.2.1. Probabilidad a priori y la regla por defecto

Si se denotan las clases por A_i, {i=1, 2, ..,q}, la probabilidad a priori por $\mathbf{B}_i$, para la clase A_i será

$$\mathbf{B}_i = P(A_i)$$

Siempre es posible utilizar la regla por defecto: clasificar cualquier observación nueva como de clase A_d, siendo A_d la de mayor probabilidad a priori, sin tener en cuenta los atributos de esa observación. Esta regla por defecto o sin datos debe ser utilizada en la práctica si el coste de recoger los datos es muy alto. Es decir, los bancos deben dar créditos a todos sus clientes habituales para mantener buenas relaciones con los clientes, en este caso el coste de recoger los datos es el riesgo de perder clientes. La regla por defecto se basa en el conocimiento de las probabilidades a priori y claramente la regla de decisión que tiene mayor oportunidad de éxito es asignar cada nueva observación a la clase más frecuente. Sin embargo, si algunos errores de clasificación son menos importantes que otros, se adoptará la regla que minimice el riesgo (menor coste

esperado) y la clase *d* es aquella con menor coste esperado, como se verá a continuación.

2.2.2. Separación de las clases

Supóngase que se observa el dato *x* en un individuo y que se sabe que la distribución de probabilidad de *x* en cada clase A_i es $P(x/A_i)$. Entonces para cada dos clases A_i y A_j el cociente de probabilidades $P(x/A_i) / P(x/A_j)$ proporciona la forma teórica óptima para discriminar las clases en base al dato *x*. Muchas de las técnicas que se utilizan pueden verse como derivaciones implícitas o explícitas y más o menos aproximadas de este cociente de probabilidades, más conocida como razón de verosimilitudes.

2.2.3. Los costes de una clasificación errónea

Supóngase que el coste de clasificar un individuo de la clase A_i como de la clase A_j es *c(i, j)*. Las decisiones se deben tomar de tal forma que se minimice el coste total de una clasificación errónea para una nueva observación, esto supone minimizar el coste esperado o coste medio de una mala clasificación.

Considérese primero el coste esperado de aplicar la regla por defecto: asignar todas las nuevas observaciones a la clase A_d , usando el subíndice *d* como etiqueta para la clase que se decide. Cuando se asigna A_d a todos los nuevos ejemplos, se incurre en un coste *c(i, d)* para los ejemplos de clase A_i y esto ocurre con probabilidad $\mathbf{B}_i$. Por tanto el coste esperado C_d de decidir A_d es:

$$C_d = \sum_{i=1}^{q} \pi_i \, c(i, d) \qquad \textbf{(2.1)}$$

La regla de coste mínimo de Bayes elige aquella clase con un menor coste esperado. Para ver la relación entre las reglas de error mínimo y las de coste mínimo, supóngase que el coste de clasificación errónea sea el mismo para todos los errores y cero cuando

una clase sea identificada correctamente, es decir, supóngase que $c(i,j) = c$ para $i...j$ y $c(i,j) = 0$ para $i=j$. Entonces el coste esperado es:

$$C_d = \sum_i \pi_i \, c(i, d) = \sum_{i \neq d} \pi_i \, c = c \sum_{i \neq d} \pi_i = c \, (1 - \pi_d) \qquad \textbf{(2.2)}$$

y la regla de coste mínimo es asignar a la clase con la mayor probabilidad a priori.

Los costes de equivocarse al clasificar son difíciles de obtener en la práctica, incluso cuando está claro que existen grandes diferencias en las penalizaciones o los premios por equivocarse o acertar. Normalmente pueden variar de un individuo a otro, como en el caso de aplicaciones para créditos de cantidades distintas en circunstancias muy diferentes. Se ha asumido que los costes de clasificación errónea son iguales para todos los individuos. En la práctica las compañías que conceden créditos deben asegurarse de los costes potenciales para cada solicitante y, en este caso, el algoritmo de clasificación normalmente da una valoración de probabilidad y la decisión se deja a un operador humano.

2.3. LA REGLA DE BAYES A POSTERIORI

Se puede ver ahora como se deben combinar los tres componentes introducidos anteriormente en un procedimiento de clasificación.

Cuando se da la información x sobre un individuo la situación es, en principio, la misma que cuando no hay datos. La diferencia es que todas las probabilidades deben interpretarse como condicionadas por el dato x. De nuevo, la regla de decisión con menor probabilidad de error es asignar a la clase con mayor probabilidad de ocurrencia, pero ahora la probabilidad relevante es la probabilidad condicionada $P(A_i / x)$ de la clase A_i dado el dato x.

$$P(A_i /x) = \text{Prob (clase } A_i \text{ , dado } x)$$

Si se quiere utilizar una regla que minimice el coste, primero se debe calcular el coste asociado a las diferentes decisiones condicionadas por la información *x*.

Ahora bien, cuando se toma la decisión A_d para ejemplos con atributos *x*, se incurre en un coste de *c(i, d)* para los ejemplos de clase A_i y esto ocurre con probabilidad $P(A_i/x)$. Como las probabilidades $P(A_i/x)$ dependen de *x*, también lo hará la regla de decisión, y en consecuencia el coste esperado $C_d(x)$ de tomar la decisión A_d .

$$C_d(x) = \sum_i P(A_i/x)\, c(i,d) \qquad \textbf{(2.3)}$$

En el caso particular de costes de clasificación errónea iguales, la regla que minimice el coste es asignar la clase con mayor probabilidad a posteriori.

Cuando se utiliza el Teorema de Bayes para calcular las probabilidades condicionadas $P(A_i/x)$ de las clases, se hace referencia a ellas como las probabilidades a posteriori de las clases. Entonces las probabilidades a posteriori $P(A_i/x)$ se calculan conociendo las probabilidades a priori π_i y las probabilidades condicionadas $P(x/A_i)$ de los datos para cada clase. Esto supone para la clase A_i que la probabilidad de observar los datos *x* es $P(x/A_i)$. El Teorema de Bayes da la probabilidad a posteriori $P(A_i/x)$ para la clase A_i como:

$$p(A_i/x) = \frac{\pi_i P(x/A_i)}{\sum_j \pi_j P(x/A_j)} \qquad \textbf{(2.4)}$$

El denominador es común para todas las clases, por tanto, a la hora de buscar la clase con menor coste esperado, se puede utilizar el hecho de que $P(A_i/x)$ es proporcional a $\pi_i\, P(x/A_i)$. Sustituyendo en la ecuación 2.3 , la clase A_d con menor coste esperado (menor riesgo) es aquella para la cual $\sum_i \pi_i\, c(i,d)\, P(x/A_i)$ es mínimo.

Todo lo anterior es igualmente válido si se trabaja con atributos que tienen distribuciones continuas, en ese caso las probabilidades anteriores se convierten en funciones de densidad de probabilidad. Suponiendo que las observaciones obtenidas de la clase A_i tienen como función de densidad de probabilidad $f_i(x) = f(x/A_i)$ y que la probabilidad a priori de que una observación pertenezca a la clase A_i es

$$p(A_i/x) = \pi_i f_i(x) / \sum_j \pi_j f_j(x) \qquad \textbf{(2.5)}$$

Una regla de clasificación entonces asigna x a la clase A_d con probabilidad a posteriori máxima dado x, o sea

$$p(A_d/x) = \max_i p(A_i/x) \qquad \textbf{(2.6)}$$

y como en el caso anterior, la clase A_d con mínimo coste esperado (mínimo riesgo) es aquella para la cual $\sum_i \pi_i c(i,d) f_i(x)$ es mínimo.

Considerando el problema de discriminar sólo entre dos clases A_i y A_j y asumiendo como antes que $c(i, i) = c(j, j) = 0$, se debe asignar a la clase i si

$$\pi_j c(i,j) f_j(x) > \pi_i c(j,i) f_i(x) \qquad \textbf{(2.7)}$$

o equivalentemente si

$$\frac{f_i(x)}{f_j(x)} < \frac{\pi_j c(i,j)}{\pi_i c(j,i)} \qquad \textbf{(2.8)}$$

que muestra el papel de pivote del ratio de probabilidades, que debe ser menor que el ratio de probabilidades a priori por el coste relativo de los errores. Se puede destacar la simetría de la expresión anterior, ya que cambios en los costes pueden compensarse con cambios en las probabilidades a priori para mantener constante el umbral que define la regla de clasificación. Esta facilidad se utiliza en algunas técnicas, aunque para más de

dos grupos esta propiedad sólo existe bajo supuestos restrictivos (ver Breiman et al 1984, pág 112).

2.3.1. La regla de Bayes en estadística

En lugar de obtener $p(A_i /x)$ vía teorema de Bayes, se puede también utilizar la versión frecuencial empírica de la regla de Bayes que, en la práctica, requerirá cantidades demasiado grandes de datos. Sin embargo, en principio, el procedimiento consiste en recoger juntos todos los ejemplos del conjunto de entrenamiento que tengan exactamente los mismos atributos que el ejemplo dado y encontrar las proporciones de las clases $P(A_i /x)$ en estos ejemplos. La regla de error mínimo es asignar la clase A_d con mayor probabilidad a posteriori.

A menos que el número de atributos sea muy pequeño y el conjunto de entrenamiento muy grande, será necesario usar una aproximación para estimar las probabilidades a posteriori de las clases. Por ejemplo, un camino para encontrar una regla de Bayes aproximada sería utilizar no sólo los ejemplos cuyos atributos coinciden exactamente con los del ejemplo dado, sino todos los ejemplos que se aproximen en algún sentido al ejemplo dado. La regla de decisión de mínimo error será la que asigne a la clase con mayor frecuencia a lo largo de estos ejemplos coincidentes. Los algoritmos de partición y los árboles de decisión en particular, dividen el espacio de los atributos en regiones de autosemejanza, todos los datos dentro de una determinada partición son tratados como similares y las probabilidades a posteriori son constantes dentro de esa partición.

Las reglas basadas en la regla de Bayes son óptimas, es decir, ninguna otra regla tiene menor ratio de error esperado o menor coste esperado. Aunque inalcanzables en la práctica, proporcionan la base lógica para todos los algoritmos estadísticos. Son inalcanzables porque asumen que existe información completa sobre las distribuciones estadísticas de cada clase.

Los procedimientos estadísticos tratan de superar la carencia de información de varias formas, pero hay dos líneas principales: *paramétricas* y *no paramétricas*. Los métodos *paramétricos* hacen supuestos sobre la naturaleza de las distribuciones (generalmente se supone que las distribuciones son Normales) y el problema queda reducido a estimar los parámetros de las distribuciones (medias y varianzas en el caso de las Normales) Los métodos *no paramétricos* no hacen supuestos sobre la distribución específica en cuestión y, por tanto, quizás son descritos de forma más precisa como métodos de *distribución libre*.

2.4. CLASIFICACIÓN DE LOS PROCEDIMIENTOS DE CLASIFICACIÓN

Los procedimientos comentados en el tema anterior (discriminante lineal, árboles de decisión y basados en reglas, vecino más próximo) son prototipos de 3 tipos de procedimientos de clasificación. Como es lógico, han sido refinados y extendidos, pero actualmente todavía representan las ramas más importantes en clasificación, tanto a nivel aplicado como a nivel de investigación. Los procedimientos que se citan a continuación pueden relacionarse directamente a uno u otro de los anteriores. Sin embargo, tradicionalmente se han dividido en 4 grupos, estadística clásica, técnicas estadísticas modernas, aprendizaje automático y redes neuronales. Para algunos de los métodos la inclusión en uno u otro grupo resulta un poco arbitraria.

2.4.1. Extensiones del discriminante lineal

Se pueden incluir en este grupo aquellos procedimientos que empiezan con combinaciones lineales de las medidas, incluso si estas combinaciones están sujetas después a transformaciones no lineales. Se pueden citar 7 procedimientos de este tipo: discriminante lineal, discriminante logístico, discriminante cuadrático, perceptrón multicapa (backpropagation y cascade), DIPOL92 y projection pursuit. Nótese que este grupo contiene métodos estadísticos y redes neuronales (concretamente perceptrón multicapa)

2.4.2. Árboles de decisión y métodos basados en reglas

Incluye gran cantidad de procedimientos entre los que destacan: NewID, AC^2, Cal5, CN2, C4.5, CART, IndCART, árbol de Bayes e Itrule.

2.4.3. Estimaciones de la densidad

Este grupo es un poco menos homogéneo, pero sus miembros tienen en común, que el procedimiento está muy relacionado con la estimación de la densidad local de probabilidad en cada punto del espacio muestral. Se puede citar: vecino más próximo, funciones de base radial, Bayes simplificado, árboles múltiples, mapas autoorganizados de Kohonen, aprendizaje por vectores de cuantificación (LVQ). Relacionado con los anteriores, el método de estimación de la densidad de núcleos, aunque a diferencia de ellos, aquí se estima la densidad de probabilidad para un núcleo. Este grupo también contiene sólo métodos estadísticos y redes neuronales.

2.5. LA SELECCIÓN DE VARIABLES

En algunos casos, se puede llegar a tener muchas características que describan a cada ejemplo y puede que no todas ellas sean igual de útiles. Por tanto, en estos casos será conveniente realizar previamente una selección de variables que presenta las siguientes ventajas.

2.5.1. Ventajas de la selección de variables

Entre las ventajas que presenta la selección de atributos se puede destacar las siguientes:

1. Reducción del tamaño del conjunto de entrenamiento.

Si se considera un conjunto de entrenamiento con *n* ejemplos, cada uno de los cuales está descrito por *q* atributos, su tamaño será del orden de *nxq*. Por tanto, para reducir el conjunto de entrenamiento se puede o bien disminuir el número de ejemplos (lo que en principio no parece muy recomendable) o bien reducir el número de atributos que describen cada ejemplo. Por tanto, se consigue reducir la información a utilizar para calcular el clasificador.

2. Velocidad en el aprendizaje.

Cuanto menor sea el número de atributos o variables utilizados para describir un problema, menor será el tiempo necesario para el aprendizaje del sistema de clasificación que se vaya a utilizar. Hay que tener en cuenta que en la mayoría de los casos el proceso de aprendizaje sólo se realiza una vez y, por tanto, será más importante reducir el tiempo necesario para clasificar una observación nueva, ya que la clasificación hay que realizarla cada vez que se presente una nueva observación. De todas formas, si el proceso de aprendizaje implica demasiado tiempo puede resultar poco práctico para aplicaciones reales.

3. Precisión en la generalización.

En algunos casos la precisión de la generalización puede mejorar al eliminar los atributos irrelevantes, o lo que es lo mismo, seleccionar a los atributos relevantes ya que disminuyen las posibilidades de que existan errores de medida en algún atributo, solapaciones o incongruencias. Además, si se seleccionan aquellos atributos que son realmente relevantes, será más fácil ajustar la verdadera función subyacente al problema en cuestión en lugar de la distribución muestral.

4. Aumento de la velocidad de clasificación.

Otra cuestión importante es la rapidez en la clasificación. Si se reduce el número de atributos utilizados se consigue (normalmente) reducir también el tiempo empleado en

la búsqueda a lo largo de los ejemplos utilizados para clasificar una nueva observación, aunque esto es especialmente importante en el aprendizaje basado en casos, por ejemplo la regla del vecino más próximo, cualquier clasificador será tanto más rápido cuantos menos valores se tengan que comprobar para clasificar una nueva observación.

Estas son las principales ventajas que se derivan de una correcta selección de atributos, para ello generalmente se habla de seleccionar los atributos relevantes o lo que es lo mismo eliminar aquellos atributos irrelevantes. Por tanto parece adecuado definir previamente qué se entiende por relevancia.

2.5.2. ¿Qué es la relevancia?

Como se acaba de decir, antes de abordar los diferentes métodos que seleccionan los atributos en función de su relevancia, se tiene que definir este concepto. En John, Kohavi y Pfleger (1994) y en Blum y Langley (1997) se pueden encontrar una serie de definiciones de este concepto. La razón de que existan diferentes definiciones para decir cuándo un atributo es relevante se puede encontrar en la pregunta ¿relevante para qué? de tal forma que en función de cuál sea el objetivo en cada caso puede existir una definición más apropiada que otra.

Definición 1 (Relevante para una clase)

Un atributo X_i es relevante para una clase C si existen 2 ejemplos A y B en el espacio de ejemplos tales que A y B sólo se diferencian en el valor que toman en el atributo X_i y C(A)...C(B).

Es decir, un atributo X_i es relevante si existe una observación en el espacio de ejemplos para el cual si se cambia el valor de X_i afecta a la clase asignada por el clasificador.

Definición 2 (Relevancia fuerte para la muestra/distribución)

Un atributo X_i es fuertemente relevante para la muestra S si existen dos 2 ejemplos A y B en S tales que A y B difieren sólo en el valor que toman en el atributo X_i y tienen diferentes clases (o diferentes distribuciones de clases si aparecen en S muchas veces). Similarmente, X_i es fuertemente relevante para la clase C y la distribución D si existen 2 ejemplos A y B que tienen probabilidad superior a cero sobre D y difieren sólo en el valor que toman en X_i y satisfacen que C(A)…C(B).

Esta definición se diferencia de la primera en que A y B ahora tienen que pertenecer a S (o tener probabilidad no nula)

Definición 3 (Relevancia débil para la muestra/distribución)

Un atributo X_i tiene relevancia débil para la muestra S (o para la clase C y la distribución D) si al eliminar un subconjunto de los atributos, X_i adquiere relevancia fuerte.

A partir de estas definiciones, interesará retener todos los atributos fuertemente relevantes ya que si se elimina un atributo de fuerte relevancia se estará creando o añadiendo ambigüedad al conjunto de entrenamiento. Un atributo de relevancia débil puede interesar retenerlo o no, según cual sean los atributos a eliminar, porque si se eliminan aquellos atributos que lo convierten en un atributo de fuerte relevancia se mantendrá, mientras que si los otros atributos no son eliminados no será necesario mantener el atributo en cuestión.

Desde otro punto de vista, en algunos casos en lugar de estudiar la relevancia individualmente para cada atributo puede interesar utilizar la relevancia como una medida de la complejidad del modelo o función que se utiliza para clasificar. En lugar de seleccionar explícitamente un subconjunto de los atributos, lo que interesa es que ese

clasificador obtenga buenos resultados cuando la cantidad de atributos sea lo más pequeña posible. Por ello se plantea la siguiente definición.

Definición 4 (Relevancia como medida de complejidad)

Dado un conjunto de ejemplos S y un conjunto de clases C, se llama r(S,C) al número de atributos relevantes utilizando la definición 1 para una de las clases de C tal que de todos aquellos cuyo error sobre S es mínimo, tiene el menor número de atributos relevantes.

Es decir, se busca el menor número de atributos necesario para lograr un comportamiento óptimo sobre S para una clase de C.

Para lograr una reducción más agresiva en el número de atributos a considerar, se pueden permitir clases con un error ligeramente superior al mínimo sobre S, si consigue un conjunto más pequeño de atributos relevantes.

En las definiciones anteriores se ha estudiado la relevancia de los atributos dejando al margen el método de clasificación a utilizar, pero en realidad nada asegura que un atributo que según estas definiciones es relevante, sea necesariamente útil para el método de clasificación que se vaya a utilizar, esto justifica la siguiente definición.

Definición 5 (Utilidad incremental)

Dado un conjunto de ejemplos S, un clasificador L y un conjunto de atributos A, se dice que el atributo X_i es incrementalmente útil para L con respecto a A si la precisión del clasificador L utilizando el conjunto de atributos $A \cup \{X_i\}$ es mejor que la precisión conseguida utilizando sólo el conjunto de atributos A.

Esta definición es muy apropiada para los métodos de selección de atributos que recorren el espacio de subconjuntos de los atributos incrementalmente, añadiendo o eliminando atributos al conjunto que acaban de utilizar (stepwise selection).

Para ver de forma más clara los aspectos principales de las definiciones anteriores, así como las diferencias entre ellas se puede observar la tabla 2.1.

Aunque se ha estudiado la relevancia de los atributos de forma individual, este enfoque puede ampliarse considerando la relevancia para combinaciones lineales de atributos, un ejemplo muy común de estos métodos es el análisis de componentes principales cuyo estudio escapa de los objetivos de este trabajo pero que puede encontrarse en cualquier manual estadístico de análisis multivariante (Por ejemplo, Análisis de datos: series temporales y análisis multivariante. E. Uriel Jiménez).

Tipo de relevancia	Definición
Relevancia para una clase	Un atributo X_i es relevante para una clase C si existen 2 ejemplos A y B *en el espacio de ejemplos* tales que A y B sólo se diferencian en el valor que toman en el atributo X_i y C(A)…C(B).
Relevancia fuerte para la muestra o distribución	Un atributo X_i es fuertemente relevante para la muestra S si existen dos 2 ejemplos A y B *en S* tales que A y B difieren sólo en el valor que toman en el atributo X_i y tienen diferentes clases. Similarmente, X_i es fuertemente relevante para la clase C y la distribución D si existen 2 ejemplos A y B que tienen probabilidad superior a cero sobre D y difieren sólo en el valor que toman en X_i y satisfacen que C(A)…C(B).
Relevancia débil para la muestra o distribución	Un atributo X_i tiene relevancia débil para la muestra S (o para la clase C y la distribución D) *si al eliminar un subconjunto de los atributos, X_i adquiere relevancia fuerte.*
Relevancia como medida de complejidad	Dado un conjunto de ejemplos S y un conjunto de clases C, se llama r(S,C) al número de atributos relevantes utilizando la definición 1 para una de las clases de C tal que *de todos aquellos cuyo error sobre S es mínimo, tiene el menor número de atributos relevantes.*
Utilidad incremental	Dado un conjunto de ejemplos S, un clasificador L y un conjunto de atributos A, se dice que el atributo X_i es incrementalmente útil para L con respecto a A *si la precisión del clasificador L utilizando el conjunto de atributos $A \subset \{X_i\}$ es mejor* que la precisión conseguida utilizando sólo el conjunto de atributos A.

Tabla 2.1. Definiciones de relevancia.

2.5.3. Selección de atributos como búsqueda heurística

Si se entienden los métodos de selección de atributos como un proceso de búsqueda heurística, donde cada estado en el espacio de búsqueda recoge un posible subconjunto de los atributos, entonces hay cuatro aspectos que determinan la naturaleza del proceso de búsqueda. En concreto en Langley (1994) se destacan las cuatro siguientes:

1. En primer lugar se debe determinar *el punto de partida* en el espacio de búsqueda lo que a su vez influirá en la dirección de la búsqueda y en la forma de generar los sucesivos estados. Estos estados se pueden ordenar en función del número de atributos que incluyan, por lo que la búsqueda puede comenzar desde cero e ir añadiendo atributos sucesivamente, o considerarlos todos al principio e ir eliminando. Estos métodos se conocen respectivamente como selección hacia delante y eliminación hacia atrás[2].

2. Una segunda característica es *la organización de la búsqueda*. Es demasiado costoso analizar todos los posibles subconjuntos de los atributos, porque existen 2^a posibles subconjuntos de *a* atributos. Una posible solución es un método gradual para recorrer el espacio de los subconjuntos. En cada punto de la búsqueda, se consideran cambios locales al subconjunto de ese momento y se prueba con uno de ellos.

El método de selección o eliminación por pasos (stepwise selection or elimination) considera en cada punto de decisión, tanto la inclusión como la eliminación de atributos, lo que permite rectificar una decisión anterior sin retroceder explícitamente en el camino de búsqueda. Dentro de estas opciones, se pueden considerar todas las combinaciones generadas y luego escoger la mejor, o se puede escoger simplemente el primer subconjunto que mejore la precisión del conjunto actual.

[2] Estos métodos se conocen en inglés como forward selection y backward elimination, respectivamente.

3. *Cómo evaluar* los distintos subconjuntos posibles de los atributos. Lo que buscan es medir la habilidad de los atributos para discriminar entre las clases del conjunto de entrenamiento. Esto se puede hacer sobre el propio conjunto de entrenamiento (acierto en reescritura) o bien sobre un conjunto de evaluación específico.

4. Por último, se debe fijar un *criterio para detener la búsqueda*. Se puede detener la búsqueda cuando ninguna de las alternativas mejore la precisión de la clasificación, o bien continuarla para reducir el número de atributos mientras que la precisión no disminuya. También se puede recorrer el espacio de búsqueda de un extremo a otro y después seleccionar la mejor combinación. Otra posibilidad es ordenar los atributos según alguna medida de relevancia y utilizar un sistema paramétrico para determinar el punto de ruptura.

Estas cuatro características que se han enunciado ayudan a diferenciar los métodos que se van a estudiar a continuación, distinguiendo dos grandes grupos, por un lado aquellos que consideran la relevancia como un concepto relativo y atribuyen pesos a los atributos en función de su relevancia y por otro los métodos mas radicales que eliminan aquellos atributos que consideran irrelevantes o lo que es lo mismo sólo utilizan los relevantes. Dentro de estos últimos se va a distinguir a su vez tres grupos: los que forman parte del propio método de clasificación que se conocen en inglés como *embedded approaches*, los que eliminan los atributos irrelevantes antes de iniciar el proceso de entrenamiento del clasificador (filter approaches) y aquellos que utilizan el método de clasificación para valorar los distintos subconjuntos de atributos de forma sucesiva (wrapper approaches).

2.5.4. Métodos incrustados de selección de atributos

La inclusión o eliminación de atributos constituye el núcleo central de los métodos que inducen conceptos lógicos, aunque algunos de éstos métodos incluyen rutinas para combinar los atributos en descripciones más importantes.

Por ejemplo, los métodos de particiones recursivas para inducir árboles de clasificación (ID3 y C4.5 de Quinlan y CART de Breiman), realizan una búsqueda gradual a lo largo del espacio de árboles de decisión, utilizando en cada paso una función de evaluación para seleccionar el atributo que tiene la mayor capacidad discriminante entre las clases. Dividiendo el conjunto de entrenamiento en base a ese atributo y repiten el proceso en cada subconjunto, desarrollando el árbol hasta que no se pueda discriminar más.

Los métodos de "separa-y-vencerás" (separate-and-conquer) incluyen selección de atributos de forma similar. Utilizan una función de evaluación para encontrar un atributo que ayude a discriminar entre la clase C y el resto, ese atributo se añade a una regla de tipo conjuntivo para C. Se añaden atributos a esa regla hasta que excluya a los miembros de otras clases, entonces se eliminan los ejemplos de la clase L que cubra la regla y se repite el proceso en los ejemplos de entrenamiento restantes.

2.5.5. Métodos de filtrado para la selección de atributos

En este caso la selección de atributos se realiza en un proceso específico para ese propósito que tiene lugar antes de realizar el proceso de inducción o aprendizaje. El nombre de método de filtrado corresponde a John, Kohavi y Pfleger (1994) y se debe a que este método elimina o filtra, los atributos irrelevantes antes de que se produzca el aprendizaje. Después de seleccionar los atributos más relevantes se puede aplicar cualquiera de los métodos de clasificación.

El método de filtrado más sencillo es comprobar el poder discriminante de cada atributo individualmente y elegir los k atributos más decisivos. La elección óptima de k se puede realizar probando en un conjunto de test.

Otra posibilidad algo más complicada es escoger la combinación mínima de atributos que discrimina perfectamente entre las clases. Para ello empieza con cada atributo por separado, luego busca entre los diferentes pares de atributos, después entre las ternas y

así sucesivamente, hasta encontrar una combinación que genere una partición “pura” del conjunto de entrenamiento, es decir que no haya ejemplos de clases distintas en una misma partición.

También pueden verse como métodos de filtrado, aquella métodos que construyen atributos de orden superior a partir de los originales, los ordenan en función de la varianza que explican y seleccionan los mejores. El análisis de Componentes Principales es el ejemplo más conocido de este tipo y genera una combinación lineal de los atributos originales cuyos vectores son ortogonales en el espacio de atributos original. Una variación de este método es el análisis de Componentes Independientes que comparte la idea principal pero exige que los nuevos atributos sean independientes en lugar de ortogonales.

2.5.6. Métodos envolventes para la selección de atributos

Estos métodos también se realizan fuera del proceso de aprendizaje propiamente dicho, pero utilizan el clasificador en cuestión como parte del proceso de selección de atributos en lugar de como un proceso posterior. El nombre de envolvente es original de John et al (1994). El espacio de subconjuntos de atributos donde se realiza la búsqueda es el mismo que para los métodos anteriores (anidado y filtrado), pero la evaluación de los subconjuntos se realiza aplicando un determinado método de clasificación sobre el conjunto de entrenamiento y utilizando la precisión del clasificador resultante para comparar los distintos subconjuntos.

El principal argumento para utilizar estos métodos envolventes es que cada método de clasificación tendrá un sesgo en la fase inductiva y, por tanto, para estimar la precisión del subconjunto de atributos a seleccionar es mejor tener en cuenta el sesgo del método de clasificación que se va a usar después y no el sesgo de otra medida distinta de precisión.

En cambio el principal inconveniente de estos métodos frente al método de filtrado es el coste computacional tan grande debido al esfuerzo que supone repetir el proceso de aprendizaje para cada subconjunto de atributos considerado.

Uno de los métodos de clasificación que habitualmente se utiliza dentro del marco de la selección de atributos envolvente, son los árboles de clasificación y clasificación por vecindad (*k*-NN). Los árboles de clasificación como se comentó anteriormente realizan dentro del propio proceso de aprendizaje una selección de atributos, por tanto, a priori parece más necesaria la selección de atributos en el caso del clasificador del vecino más próximo, ya que este sencillo método requiere almacenar gran cantidad de información para clasificar cada nueva observación. Por ello se han desarrollado los métodos envolventes para clasificar por vecindad y aprendizaje basado en ejemplos/casos.

Estos métodos envolventes de selección de atributos y en general todos los de selección de atributos, son principalmente recomendables para aquellos métodos de clasificación especialmente sensibles a los atributos irrelevantes o redundantes.

2.5.7. Métodos que ponderan los atributos

Hasta ahora se han comentado los métodos que seleccionan un subconjunto relevante de atributos. Sin embargo, existe una posibilidad menos drástica que eliminar los atributos irrelevantes y quedarnos exclusivamente con los relevantes, esta posibilidad consiste en asignar un peso a los atributos en función de su relevancia, de tal forma que a aquellos atributos con mayor relevancia les corresponderá un mayor peso. Por tanto, en lugar de diferenciar entre atributos relevantes e irrelevantes se hablará ahora de distintos grados de relevancia.

En general la selección de atributos será más adecuada para aquellos métodos de clasificación que deben ser humanamente inteligibles o cuyo resultado puede ser utilizado por un algoritmo posterior. La ponderación de los atributos es recomendable

en conjuntos que se van incrementando on-line y su propósito es únicamente aumentar la eficiencia.

De todas formas la separación entre selección de atributos y ponderación no es tan tajante ya que los tres métodos que se han visto como selección de atributos se pueden modificar para ponderar los atributos en lugar de eliminarlos o seleccionarlos. Además se pueden compatibilizar las dos visiones, utilizando un sistema que asigne pesos a los atributos en función de la relevancia, pero que al mismo tiempo elimine aquellos que no superen un umbral mínimo de dicha relevancia.

2.6. ESTIMACIÓN DE LA PRECISIÓN DE UN SISTEMA DE CLASIFICACIÓN

El objetivo de cualquier sistema de clasificación entrenado en un conjunto de ejemplos es aprender a clasificar correctamente las nuevas observaciones que se le presenten. Para evaluar el comportamiento futuro del clasificador ante nuevos ejemplos, se intentará estimar la precisión del clasificador, al mismo tiempo esta estimación ayuda a seleccionar el sistema de clasificación más apropiado en cada caso para el problema concreto (número y tipo de atributos) y para los requisitos que se le impongan al clasificador que se quiera obtener (número de reglas y/o de conjunciones, etc.).

Para llevar a cabo la estimación se utilizan ciertas técnicas estadísticas de muestreo (remuestreo aleatorio, bootstrap, etc.). Del estimador que se obtenga se estudiará su sesgo (errores no aleatoreos en la estimación, puede ser optimista o pesimista) y su variabilidad (debe ser lo más estable posible, independientemente de la muestra utilizada). Para comparar la precisión de dos clasificadores, interesa más que el estimador utilizado sea estable, ya que si está sesgado de manera pesimista u optimista afectará en el mismo sentido a ambos, por lo que no entorpece la comparación.

Para medir la bondad de un clasificador se utiliza su tasa o porcentaje de error. *La tasa de error real* de un clasificador se define como el porcentaje de error del clasificador cuando se prueba sobre la verdadera distribución de casos de la población,

lo que se puede aproximar empíricamente por un número muy grande de nuevos casos recogidos de forma independiente a los ejemplos utilizados para entrenar al clasificador.

La tasa de error se calcula como el cociente entre el número de errores cometidos y el número de casos examinados,

$$Tasa\ de\ error = \frac{número\ de\ errores}{número\ de\ casos} \tag{2.9}$$

La verdadera tasa de error se podría calcular si el número de ejemplos tiende a infinito. Pero en la realidad, el número de ejemplos disponibles siempre es finito y muchas veces relativamente pequeño. Por tanto, se tiene que estimar la tasa real de error a partir de la tasa de error calculada sobre una pequeña muestra. Como ya se ha dicho algunas estimaciones presentarán un sesgo, pueden ser demasiado bajas, en el caso optimista, o demasiado altas, en el caso pesimista.

Por *error* se entiende aquellos casos en que el clasificador asigna una clase incorrecta a la observación que se le ha presentado, para simplificar se supone, salvo que se indique lo contrario, que todos los errores son igual de importantes, lo que en la realidad no tiene porque ser así.

2.6.1. Tasa de error aparente

La tasa de error aparente de un clasificador es el porcentaje de error del clasificador en los ejemplos que se han utilizado para construirlo o entrenarlo. Esta tasa de error aparente también se conoce como tasa de error en resustitución o en reescritura.

Si el conjunto de entrenamiento fuese ilimitado la tasa de error aparente coincidiría con la tasa de error real, pero como ya se ha apuntado esto no sucede en el mundo real y en general se tendrán muestras de tamaño limitado, más bien pequeño, a partir de las cuales se construirá el clasificador y se estimará su comportamiento sobre nuevos casos.

Para la mayoría de los clasificadores la tasa de error en reescritura es un mal estimador de su futuro comportamiento. En general, la tasa de error aparente estará sesgada de forma optimista y por tanto será menor que la verdadera tasa de error. En especial, cuando el clasificador ha sido sobreajustado o sobreespecializado a las características particulares de la muestra en lugar de recoger la estructura subyacente a la población. Esto puede llevar a clasificadores con una tasa de error aparente muy baja, incluso puede llegar a ser cero, es decir con un comportamiento muy bueno en los ejemplos del conjunto de entrenamiento o aprendizaje, pero cuyo comportamiento sea muy malo ante nuevas observaciones. Se dice entonces que el clasificador está sobreajustado o sobreespecializado porque pierde precisión a la hora de generalizar.

2.6.2. Estimación de la tasa de error real

Aunque la tasa de error aparente es normalmente engañosa, existen diversas formas de intentar estimar el error real. El requisito para que cualquier estimación pueda ser correcta, es que la muestra o conjunto de datos debe ser una muestra aleatoria, ya que si la muestra ha sido preseleccionada puede que no recoja la estructura subyacente a la población y por tanto el clasificador obtenido perderá precisión ante observaciones nuevas.

En la práctica, la mayoría de las veces el conjunto de datos no ha sido obtenido de forma aleatoria, ya que en el proceso de recogida de la información no se han tenido en cuenta los métodos estadísticos de muestreo o simplemente porque no ha existido la oportunidad de aplicarlos, se ha recogido toda la información disponible y aún así el conjunto de datos es muy pequeño. En función del tamaño de datos se tienen dos formas distintas para estimar el error real, la estimación por conjunto de prueba y las técnicas de remuestreo.

Cuando el tamaño del conjunto de datos lo permita, la forma más sencilla para estimar el error real será la *estimación por conjunto de prueba*, este método divide el

conjunto de datos aleatoriamente en dos conjuntos, uno se utiliza para entrenar o construir el clasificador y el otro para probar su precisión.

Sin embargo, para muestras pequeñas o de tamaño moderado es recomendable utilizar alguna de las técnicas de remuestreo (submuestreo aleatorio, validación cruzada, leaving one out y bootstrapping) que básicamente lo que hacen es repetir muchas veces la división aleatoria entre conjunto de entrenamiento y de prueba.

Todos estos métodos se utilizan para estimar la precisión del clasificador, pero el clasificador final se calculará sobre todo el conjunto de datos disponibles.

2.6.3. Estimación por conjunto de prueba

Como se ha visto, en muchas ocasiones el número de ejemplos disponibles para entrenar el clasificador y estimar su precisión es limitado, por tanto es interesante dividir el conjunto original en dos grupos: el conjunto de entrenamiento y el conjunto de prueba. De tal forma que el clasificador se diseña utilizando exclusivamente el conjunto de entrenamiento, escondiendo o dejando al margen los ejemplos del conjunto de prueba, de ahí que este método de estimación se conozca en inglés como Hold out. Una vez que el clasificador está construido, entonces se le presentan los ejemplos de prueba para que los clasifique como si fuesen nuevas observaciones, obteniendo así una estimación del comportamiento del clasificador ante nuevas observaciones. El porcentaje de error del clasificador en los ejemplos de prueba, se llama *tasa de error en el conjunto de prueba*.

La división en ambos conjuntos debe ser aleatoria, normalmente se establece un porcentaje fijo de ejemplos a utilizar para entrenar y el resto para probar el clasificador obtenido. Las proporciones normalmente son de 2/3 y 1/3, ya que el proceso de entrenamiento es más importante la mayoría de ejemplos se utilizan para entrenar. Sin embargo, para grandes conjuntos, una vez que el número de ejemplos de prueba supera los mil, se puede dedicar un mayor porcentaje al entrenamiento.

Para grandes conjuntos de ejemplos, la estimación por conjunto de prueba produce buenos resultados para el diseño y prueba del clasificador. Sin embargo, para conjuntos de un tamaño moderado, este método deja con insuficientes observaciones a uno de los dos conjuntos, el de entrenamiento o el de prueba. Las observaciones del conjunto de prueba contienen información útil para el aprendizaje pero son ignoradas durante el entrenamiento. Por tanto la estimación por conjunto de prueba (2/3 y un 1/3) es relativamente pesimista respecto al error real cuando se entrena el mismo método de clasificación sobre el conjunto de datos total.

Para solucionar el sesgo pesimista de la estimación por conjunto de prueba, se pueden utilizar las técnicas de remuestreo, que repiten sucesivas veces la partición del conjunto de datos en conjunto de entrenamiento y conjunto de prueba.

2.6.4. Submuestreo aleatorio

Este método consiste en repetir la partición aleatoria del método anterior sucesivas veces, de tal forma que cada vez se diseña un nuevo clasificador a partir del conjunto de aprendizaje que resulta en cada partición. La tasa de error estimada será la media de los errores de los clasificadores calculados sobre los conjuntos de prueba generados de manera independiente y aleatoria.

El submuestreo aleatorio soluciona el problema de tener que confiar en una única partición que puede resultar ser no representativa, promediando los resultados sobre muchas particiones entrenamiento-prueba generadas aleatoriamente. La siguiente tabla destaca las similitudes y diferencias entre este método y el anterior.

	Estimación por C. de prueba	Submuestreo aleatorio
Conjunto de entrenamiento	n_1	n_1
Conjunto de prueba	$n_2 = N - n_1$	$n_2 = N - n_1$
Iteraciones	1	$b << N$

Tabla 2.2. Comparación de la estimación por conjunto de prueba y submuestreo aleatorio.

Donde N es el número total de ejemplos disponibles, n_1 y n_2 son respectivamente el tamaño del conjunto de entrenamiento que puede variar de 1 a N y b es el número de veces que se repite el proceso que debe ser sustancialmente menor que N.

2.6.5. Validación cruzada

En realidad puede considerarse como un caso particular del submuestreo aleatorio, en este caso se divide aleatoriamente el conjunto original de datos en *k* subconjuntos mutuamente excluyentes y de aproximadamente el mismo tamaño, que en inglés se llaman folds y el método se conoce como *k*-folds cross validation. Cada uno de estos subconjuntos se utilizan como conjunto de prueba para el clasificador construido a partir de los *k*-1 subconjuntos restantes, de tal forma que al final se tendrán *k* clasificadores distintos con sus respectivas tasas de error en conjunto de prueba.

La tasa de error estimada por validación cruzada será las media de las *k* tasas de error calculadas, ponderando por los tamaños de los subconjuntos si son de distinto tamaño. En el caso de que el tamaño de todos los subconjuntos coincida, no será necesario ponderar por sus tamaños.

Este estimador es optimistamente sesgado respecto a la tasa de error real, según los resultados empíricos que aparecen en R. Kohavi (1995) si *k* es menor que cinco las estimaciones serán demasiado sesgadas, cuando *k* está próximo a diez habrá un sesgo aceptable y si *k* es mayor que 20 las estimaciones serán casi insesgadas. También se comprueba que la validación cruzada estratificada en general tiene menor sesgo.

Este método presenta frente al submuestreo aleatorio la gran ventaja de que todos los ejemplos disponibles se utilizan tanto en el aprendizaje, cosa que es fundamental, como para comprobar la precisión. El valor de *k*, es decir, el número de particiones que más se utiliza es diez, consiguiendo así que en cada prueba la reducción del conjunto de entrenamiento no sea demasiado grande, frente al 1/3 que se suele utilizar en el submuestreo aleatorio. La desventaja es el coste computacional porque repite *k* veces el proceso de aprendizaje.

En algunos casos puede ser recomendable la estratificación al realizar las k particiones, de tal forma que se mantengan las proporciones de cada clase en el conjunto total, sobre todo en términos de la jerarquía de las clases ya que podría darse el caso de que una clase mayoritaria en el conjunto original quedase relegada a un segundo plano en alguno de los subconjuntos, lo que dañaría la estimación de la tasa de error real.

Cuando el conjunto de datos es muy pequeño es aconsejable utilizar un caso particular de la validación cruzada, el leaving one out.

2.6.6. Leaving one out

Este método es un caso particular de la validación cruzada, donde el conjunto original se divide de forma aleatoria en N subconjuntos (*k*=N) de tal forma que cada uno de ellos sólo contiene una observación.

Este procedimiento es aún más costoso que la validación cruzada con 10 subconjuntos ya que en este caso hay que construir N clasificadores, cada uno de ellos a partir de N-1 ejemplos, por ello este método sólo es aconsejable para problemas con un conjunto de datos pequeño.

Para un determinado método de clasificación y un conjunto de N ejemplos, se genera un clasificador utilizando N-1 ejemplos y se prueba en el ejemplo que se ha dejado fuera (de ahí el nombre de leaving one out, ya que en cada iteración se deja un ejemplo fuera

del entrenamiento). Esto se repite para los N ejemplos construyendo cada vez el clasificador. Así cada ejemplo del conjunto inicial es utilizado una vez como ejemplo de prueba y en cada iteración casi todos los ejemplos se utilizan para construir o entrenar el clasificador.

La tasa de error estimada por leaving one out es el número de errores en los ejemplos de prueba dividido por N y constituye un estimador insesgado de la tasa de error real. En la siguiente tabla se resumen los métodos de validación cruzada (con 10 subconjuntos) y de leaving one out para un conjunto con N ejemplos.

	Validación cruzada	Leaving one out
Conjunto de entrenamiento	90%	N-1
Conjunto de prueba	10%	1
Iteraciones	10	N

Tabla 2.3. Comparación de Validación cruzada y Leaving one out.

2.6.7. Bootstrapping

Aunque el estimador leaving one out es casi insesgado, su varianza es grande para pequeñas muestras (30 casos o menos), es decir, a la larga en promedio se aproxima al error real, pero los valores individuales pueden alejarse mucho de ese error real. Para solucionar esto se utiliza el método de bootstrapping.

En la práctica económica o empresarial hay muchos casos en los que interesa aprender lo antes posible, es decir, utilizar conjuntos de ejemplos muy pequeños ya que cada nueva observación supondrá un coste, sobre todo en aquellos casos en los que una de las clases sea especialmente perjudicial. Por ejemplo, en control de calidad para decidir si un producto es defectuoso o no, cuanto antes se aprenda a clasificar, menos veces habrá que asumir el coste de un producto defectuoso. Lo mismo pasa con la clasificación de los clientes que solicitan un crédito a un banco, en morosos y no

morosos, cuantos menos morosos se tengan que sufrir hasta llegar a poder reconocerlos de antemano menor coste supondrá para la entidad. Por tanto, habrá que trabajar con muestras muy pequeñas, de treinta casos o menos y se utilizará el método de bootstrapping.

Como siempre se parte de un conjunto con N ejemplos y se selecciona para el conjunto de entrenamiento N ejemplos mediante muestreo con reemplazamiento, lo que significa que un mismo ejemplo puede estar repetido dos o más veces en un mismo conjunto de entrenamiento. Como conjunto de prueba se utilizan aquellos ejemplos que no han sido incluidos ninguna vez en el conjunto de entrenamiento.

La proporción esperada de ejemplos que pasan a formar parte del conjunto de entrenamiento es de 63,2% y la proporción esperada de ejemplos en el conjunto de prueba es de 36,8%. La tasa de error estimada es la media de las de error después de varias iteraciones. Alrededor de doscientas iteraciones se consideran necesarias, para que esta estimación sea buena por tanto este método es computacionalmente mucho más costoso que el leaving one out, la siguiente tabla recoge una comparativa de ambos métodos.

	Leaving one out	Bootstrapping
Conjunto de	N-1	N(j distintos)
Conjunto de prueba	1	N - j
Iteraciones	N	200

Tabla 2.4. Comparación de Leaving one out y Bootstrapping.

Donde *j* es el número de ejemplos distintos seleccionados para formar parte del conjunto de entrenamiento.

En Weiss y Kulikowski (1991) se recomienda el método de remuestreo a utilizar en función del número de ejemplos disponibles:

- Para tamaños muestrales superiores a cien, se recomienda utilizar validación cruzada. Aunque también se puede utilizar validación cruzada estratificada o leaving one out, el coste computacional del leaving one out es muy superior y la validación cruzada con k=10 obtiene buenos resultados.

- Para muestras con menos de cien ejemplos, es recomendable el leaving one out.

- Para muestras muy pequeñas, menos de treinta ejemplos, se recomienda el bootstrapping.

	C. de prueba	Submuestreo aleatorio	Validación cruzada	Leaving one out	Bootstrapping
C. entrenamiento	n_1	n_1	90%	N-1	N(j distintos)
C. prueba	$n_2 = N - n_1$	$n_2 = N - n_1$	10%	1	N - j
Iteraciones	1	b << N	10	N	200
Recomendable para	n_2>1000	N grande	N>100	N<100	N< 30

Tabla 2.5. Comparación de los métodos de estimación de la tasa de error real.

2.6.8. El error estándar

Aunque en los estimadores anteriores en la mayoría de los casos sólo se ha comentado su sesgo, también hay que tener en cuenta la variabilidad de los mismos para conocer su precisión. Es decir, es interesante determinar para la tasa de error calculada a partir de N ejemplos independientes, cuánto se aleja de la tasa real de error. Para esto se utiliza la desviación estándar (SD, standard deviation) o error estándar (SE, standard error). Para un conjunto de ejemplos de prueba independientes y a partir de la distribución binomial, el error estándar se calcula como

$$SE = \sqrt{\frac{\varepsilon(1-\varepsilon)}{N}} \qquad \textbf{(2.10)}$$

donde N es el número de ejemplos independientes que se utiliza como conjunto de prueba.

Cuando se utiliza alguna de las técnicas de remuestreo que se han visto, N será el número total de ejemplos. Pero la estimación del error estándar será sesgada de forma optimista, porque todos los casos de la muestra se utilizan para entrenar al clasificador y también para probarlo y, en consecuencia, los ejemplos de prueba no son completamente independientes. Esto se soluciona cuando la muestra es algo mayor, sobre 200 casos, logrando un estimador muy próximo al verdadero error estándar.

CAPÍTULO 3

MÉTODOS DE CLASIFICACIÓN INDIVIDUALES:

ANÁLISIS DISCRIMINANTE Y VECINO MÁS PRÓXIMO

3.1. INTRODUCCIÓN

Este primer capítulo dedicado a los métodos clasificación individuales, está dividido en dos partes claramente diferenciadas, la primera de ellas se centra en el análisis discriminante y la segunda en el método del vecino más próximo.

En primer lugar, se estudia el discriminante lineal, que es el procedimiento más clásico de clasificación y el más difundido entre los programas estadísticos. La idea es dividir el espacio muestral mediante una combinación lineal de las variables o atributos que describen los ejemplos. La combinación lineal determinará una recta, un plano o un hiperplano, según sea un espacio de dos, tres o más dimensiones.

A continuación se aborda el discriminante cuadrático que es muy similar al lineal, pero cuyas fronteras de decisión entre dos regiones distintas pueden ser superficies cuadráticas. Este método se utiliza cuando las matrices de covarianzas de la distribución que siguen los atributos no son iguales. Otra alternativa al discriminante lineal es el discriminante logístico que, ante cada nueva observación, busca la clase con mayor probabilidad condicionada a los valores que presenta esa observación. Sin embargo, en la práctica, a menudo las diferencias entre ambos métodos es pequeña y el discriminante lineal proporciona buenos valores de partida para el logístico.

El conjunto de entrenamiento recoge "*n*" ejemplos de las "*q*" clases conocidas (a menudo *q=2*). Para cada ejemplo se conocen los valores de "*p*" atributos numéricos que forman el vector $x = (x_1, x_2, x_3,....,x_p)$. Estos métodos requieren que los atributos sean numéricos o continuos y que no exista ningún valor desconocido. Cuando un atributo es categórico o discreto con dos clases, se sustituye por un indicador, un atributo que toma el valor 1 para una categoría y 0 para la otra. Cuando el número de posibles categorías es superior a dos, se utiliza un indicador para cada categoría y para que no exista redundancia se elimina uno de los indicadores. De esta forma, un atributo categórico con *j* valores se sustituye por *j-1* atributos cuyos valores son 0 ó 1. Será 0 para todos los indicadores salvo para el correspondiente a la modalidad del atributo que presenta el ejemplo, que toma el valor 1. Si todos son cero, los *j-1*, entonces el ejemplo presenta la modalidad del indicador eliminado. Cuando los valores del atributo están ordenados, es aceptable utilizar un atributo numérico sencillo. Hay que tener cuidado de que los números utilizados reflejen el espacio de las categorías de un modo aceptable.

En la segunda parte del capítulo, a partir del apartado 3.5, se comenta el método de clasificación del vecino más próximo empezando por el de orden uno y generalizando después al caso de los *k* vecinos más próximos. Se analiza el comportamiento de la regla de orden *k*, a medida que aumenta *k* y a medida que aumenta el tamaño del conjunto de observación, comparándola con la regla óptima de Bayes.

Se considera la posibilidad de exigir un porcentaje mínimo (umbral) a la clase a asignar, para tener una mayor seguridad especialmente cuando exista una clase mayoritaria en el conjunto de éxitos. Se exponen las principales limitaciones a la hora de aplicar este método y las posibles soluciones para superarlas.

El método del vecino más próximo se basa en la creencia de que las observaciones cercanas tienden a ser de la misma clase, esta creencia depende de que se considere únicamente al vecino más próximo a la observación a clasificar, que se suele representar por 1-NN (del inglés one nearest neighbour) o bien el caso general donde se tienen en cuenta los *k* vecinos más próximos (*k*-NN) a la nueva observación. Estas reglas proporcionan una estimación directa de la probabilidad a posteriori de cada una de las clases, a partir del número de ejemplos de cada clase presentes dentro de los k vecinos más próximos al ejemplo que se quiere clasificar. Además, las reglas de clasificación resultantes son sencillas y fáciles de entender.

En concreto la regla de clasificación basada en los k vecinos más próximos, dada una observación *x* cuya clase es desconocida, dice que se le asigne aquella clase A_d que sea la que tiene un mayor número de ejemplos entre los *k* vecinos más próximos *x*.

3.2. DISCRIMINANTE LINEAL

El análisis discriminante es una técnica que se utiliza para clasificar individuos en distintos grupos preexistentes, a partir de la información de un conjunto de variables. Estas variables se denominan *variables clasificadoras* y la información se sintetiza en lo que se denomina *funciones discriminantes*. Se trata de una técnica de dependencia, donde se establece la relación entre una única variable dependiente cualitativa (categórica cuyas clases identifican los grupos preexistentes) y un conjunto de variables independientes cuantitativas.

El análisis discriminante puede utilizarse con dos finalidades:

Finalidad descriptiva: dada una serie de variables sobre distintos individuos, se pretende caracterizar la separación entre grupos, tratando de determinar la contribución de cada una de esas variables a la separación o clasificación de los individuos.

Finalidad predictiva: cuando se plantea qué ocurre con un individuo nuevo, es decir, la asignación del individuo a uno de los grupos a la luz de los valores que toman las variables clasificadoras.

Como se comentó en el capítulo 1 existen múltiples aplicaciones, algunos ejemplos de éstas pueden ser:

- Clasificación de clientes por su riesgo de crédito, por ejemplo, clientes que han pedido un préstamo en cumplidores o fallidos, con variables clasificadoras como el nivel de ingresos, patrimonio, etc.

- Clasificación de empresas declaradas en quiebra o que funcionan sin problemas atendiendo a la información de un conjunto de variables económico financieras.

- Clasificación de productos que puedan tener éxito o fracaso en sus ventas en función de la valoración de un conjunto de características de los mismos como la duración, calidad, precio, diseño, etc.

En definitiva, el Análisis Discriminante tiene aplicación en cualquier problema de investigación cuyo objetivo sea el estudio de pertenencia a grupo de un conjunto de individuos, bien persiguiendo la clasificación de los mismos o la mejor comprensión de la discriminación entre grupos.

Para ilustrar mejor el procedimiento, se utiliza el caso simplificado en el que existen dos grupos o poblaciones (I y II) y un conjunto de individuos a los que se quiere

clasificar. Para realizar la asignación se dispone de información relativa a dos variables clasificadoras (X_1 y X_2).

Si se representan gráficamente las dos variables de forma conjunta, se obtendría una nube de puntos para cada uno de los grupos.

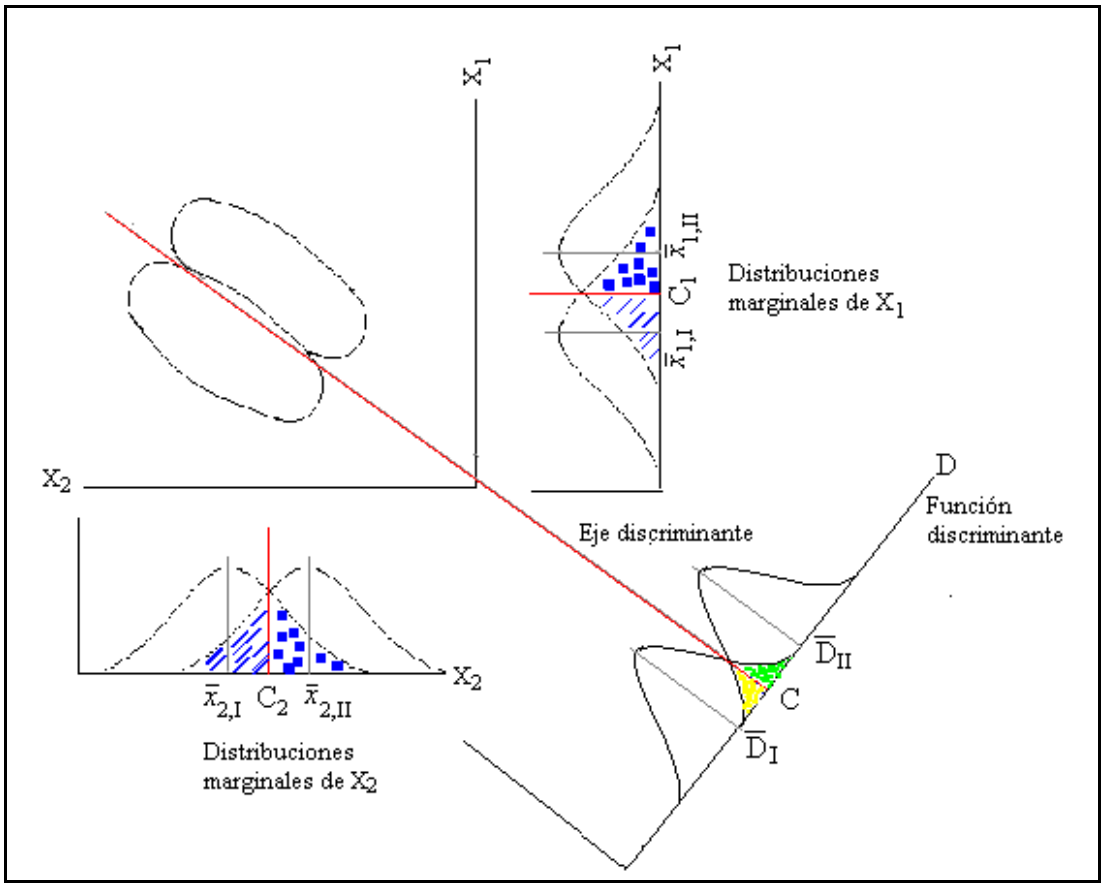


Figura 3.1 Elipses de concentración de las distribuciones de frecuencias y su proyección sobre los ejes X_1 y X2 y sobre el eje discriminante.

Mediante la función lineal (D) que se obtiene como combinación lineal de las variables de partida se puede utilizar conjuntamente toda la información de la que se dispone. Esto se puede generalizar al caso de más de dos grupos y más de dos variables discriminatorias.

3.2.1. Obtención de las ecuaciones de las funciones discriminantes

Si se parte de *p* variables (X_1, X_2, X_3,....,X_p) medidas sobre "*N*" ejemplos de "*q*" clases o grupos, la forma de las funciones discriminantes es la siguiente:

$$d_{km} = u_1 x_{1\,km} + u_2 x_{2\,km} + ...+ u_p x_{p\,km} \qquad \textbf{(3.1)}$$

donde:

d_{km} es el valor de la función discriminante para el caso m en el grupo *k*.

$x_{i\,km}$ es el valor de la variable discriminatoria X_i, expresado en desviaciones respecto a su media, para el caso m en el grupo *k*.

u_i con i=1, 2,..., *p* es el conjunto de coeficientes de la función discriminante.

Estas funciones deben conseguir la máxima diferenciación posible entre clases. Para la primera función discriminante que se determina, los coeficientes u_i se obtienen de modo que maximicen la diferencia entre el valor de la función discriminante evaluada en los centroides, o valores medios de cada grupo. Para la segunda función, los coeficientes se determinan de modo que satisfagan el mismo criterio y que además estén incorrelados con los coeficientes obtenidos en la primera función. Las funciones sucesivas se obtienen de forma similar.

El número máximo de funciones que se puede llegar a determinar será igual al número de variables discriminantes o bien al número de grupos menos uno, tomando el menor de estos dos valores. Es decir, será el valor mínimo entre *q-1* y *p*.

De forma matricial se puede expresar como **D = X AU**, siendo **D** el vector columna de orden *N*, **X** la matriz de datos de orden *Nxp* y **U** el vector columna de orden *p*.

Como se pretende maximizar las diferencias entre la media de las nuevas variables *d* en los *q* grupos, se puede utilizar el cociente entre las matrices de varianzas-covarianzas intergrupos e intragrupos, obtenidas al dividir las respectivas sumas de cuadrados por los correspondientes grados de libertad.

$$\frac{\boldsymbol{B}/(q-1)}{\boldsymbol{W}/(N-q)} = \frac{\boldsymbol{B}}{\boldsymbol{W}}\frac{N-q}{q-1} \qquad \textbf{(3.2)}$$

donde **B** y **W** son las sumas de cuadrados intergrupos e intragrupos respectivamente. Como *N* y *q* son constantes para el problema en el que se esté, maximizar el cociente inicial es equivalente a maximizar el cociente de las sumas de cuadrados **B**/**W**.

Para entender mejor las matrices **B** y **W**, considérese primero la matriz total de sumas de cuadrados y productos cruzados (**T**), en la que cada elemento t_{ij} viene definido por la expresión:

$$t_{ij} = \sum_{k=1}^{q}\sum_{m=1}^{n_k}(x_{ikm}-\bar{x}_i)(x_{jkm}-\bar{x}_j) \qquad \textbf{(3.3)}$$

donde:

q es el número de grupos.

n_k es el número de casos en el grupo k.

$x_{i\,km}$ y $x_{j\,km}$ son los valores de las variable i y j para el caso m en el grupo k.

$\bar{x}_i$ y $\bar{x}_j$ son las medias totales de las variables i y j.

Si se divide t_{ij} por *N-1*, siendo n el número total de casos, se obtiene la covarianza entre las variables *i* y *j*. Cuando *i=j* se obtiene la varianza de esa variable. La matriz de sumas de cuadrados y productos cruzados intragrupos (**W**) es similar a la matriz **T**, con la única diferencia de que las desviaciones en cada variable son ahora calculadas

respecto a la media de la clase o grupo al que pertenece cada caso. Cada elemento de esta matriz podrá expresarse como,

$$w_{ij} = \sum_{k=1}^{q} \sum_{m=1}^{n_k} (x_{ikm} - \bar{x}_{ik})(x_{jkm} - \bar{x}_{jk}) \quad \textbf{(3.4)}$$

donde $\bar{x}_{ik}$ y $\bar{x}_{jk}$ son las medias de la variables *i* y *j* en el grupo *k*. La matriz **W** recoge información sobre la variabilidad en el interior de los grupos.

De la misma forma los elementos de la matriz de sumas de cuadrados y productos cruzados intergrupos (**B**), se pueden expresar como:

$$b_{ij} = \sum_{k=1}^{q} n_k (\bar{x}_{ik} - \bar{x}_i)(\bar{x}_{jk} - \bar{x}_j) \quad \textbf{(3.5)}$$

Las matrices **W** y **B** contienen por tanto la información básica acerca de las relaciones intragrupos e intergrupos para las variables y van a utilizarse para expresar las sumas de cuadrados de la nueva variable *D*, obtenida como combinación lineal de las *p* variables independientes. Ambas sumas de cuadrados quedarían como:

$$\begin{aligned} SC_{inter} &= \boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{B} \boldsymbol{U} \\ SC_{intra} &= \boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{U} \end{aligned} \quad \textbf{(3.6)}$$

donde **U** es el vector columna formado por los pesos u_1, u_2, u_3,....,u_p y $\mathbf{U}^{\mathbf{T}}$ es su traspuesto. Por tanto, si se quiere maximizar el cociente $\mathbf{SC}_{\mathbf{inter}}/\mathbf{SC}_{\mathbf{intra}}$ se tendrá que maximizar el cociente:

$$\frac{\boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{B} \boldsymbol{U}}{\boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{U}} = \lambda \quad \textbf{(3.7)}$$

donde λ es un escalar que se puede tomar como criterio para medir la discriminación de grupos a lo largo de la dimensión especificada por el vector **U**. Se trata por tanto, ahora de encontrar el valor **U** que maximice el valor de λ, para ello habrá que calcular la derivada parcial de λ respecto a **U**, e igualar a cero. El resultado será:

$$\frac{\partial \lambda}{\partial \boldsymbol{U}} = \frac{\partial(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{U}/\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U})}{\partial \boldsymbol{U}} = \frac{2\boldsymbol{B}\boldsymbol{U}(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U}) - 2\boldsymbol{W}\boldsymbol{U}(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{U})}{(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U})^{T}(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U})} = 0$$

$$2\boldsymbol{B}\boldsymbol{U}(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U}) = 2\boldsymbol{W}\boldsymbol{U}(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{U})$$

$$\frac{\boldsymbol{B}\boldsymbol{U}}{\boldsymbol{W}\boldsymbol{U}} = \frac{\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{B}\boldsymbol{U}}{\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{U}} = \lambda \qquad \textbf{(3.8)}$$

$$\boldsymbol{B}\boldsymbol{U} = \lambda \boldsymbol{W}\boldsymbol{U}$$

$$\boldsymbol{W}^{-1}\boldsymbol{B}\boldsymbol{U} = \lambda \boldsymbol{W}^{-1}\boldsymbol{W}\boldsymbol{U}$$

$$(\boldsymbol{W}^{-1}\boldsymbol{B} - \lambda \boldsymbol{I})\boldsymbol{U} = 0$$

La ecuación obtenida servirá para determinar el eje discriminante, siendo **I** la matriz identidad de orden igual al de la matriz ($\mathbf{W^{-1}\,B}$) que será el mínimo entre *q-1* y *p*. Dado que queremos maximizar la relación $\mathbf{SC_{inter}/SC_{intra}} = \mathbf{8}$ se elegirán los autovalores de la matriz ($\mathbf{W^{-1}\,B}$) de mayor a menor y $\mathbf{U} = (u_1, u_2, u_3, ..., u_p)$ será el autovector asociado a cada autovalor. Generalmente se impone además la condición de normalización para el vector **U**, expresada como:

$$\boldsymbol{U}^{\boldsymbol{T}}\left(\frac{\boldsymbol{W}}{N-q}\right)\boldsymbol{U} = 1 \qquad \textbf{(3.9)}$$

Los coeficientes u_i cuando se calculan directamente a partir de los valores de los individuos en las *p* variables, reciben el nombre de coeficientes no estandarizados. Mientras que si los valores de los individuos son tipificados antes de calcular los coeficientes, los coeficientes obtenidos se llamarán coeficientes estandarizados.

Es interesante calcular la importancia relativa de cada función discriminante. El primer autovalor ($\mathbf{8}_1$) indica el poder discriminante de la primera función discriminante, el segundo autovalor el de la segunda función discriminante, y así sucesivamente. Por tanto, la importancia relativa de la función discriminante *i*-ésima se puede expresar como:

$$\frac{\lambda_i}{\sum_{j}^{\min(q-1,p)} \lambda_j} \qquad \textbf{(3.10)}$$

El primer eje será el que más discrimine, el segundo discriminará menos que el primero pero más que el resto, y así sucesivamente.

3.2.2. Contrastes en el análisis discriminante

Si ahora se realizan supuestos sobre la población teórica de la que ha sido extraída la muestra se puede llegar más allá en el análisis y realizar, por ejemplo, tests para contrastar la capacidad de discriminación del modelo, contrastes para seleccionar las variables que deben ser seleccionadas en el análisis y llevar a cabo cálculos de probabilidad de pertenencia a cada grupo.

Las hipótesis que se realizan se refieren a la composición de los grupos. Se considera cada grupo como una población sobre la que se establecerán los siguientes supuestos:

1.- Hipótesis de homoscedasticidad u homogeneidad de las matrices de varianzas-covarianzas.

$$\mathbf{E}_k = \mathbf{E} \quad ; \forall k = 1,2,..., q$$

2.- Hipótesis sobre la obtención de la muestra. La matriz de datos constituye una muestra aleatoria multivariante independiente en cada uno de los grupos. Bajo este supuesto, la función discriminante de Fisher es óptima.

a) Contraste de significación

En primer lugar se puede contrastar si las medias poblacionales de cada grupo son estadísticamente diferentes. Es decir, en el caso general de *q* clases se contrasta

$$H_0: \mu_1 = \mu_2 = ... = \mu_q$$

$$H_1: \mu_1 \dots \mu_2 \dots\dots\dots \mu_q$$

Para ello se utiliza el estadístico *Lambda de Wilks* (Λ), que se define como el cociente entre el determinante de la matriz de variabilidad intragrupos y el de la matriz de variabilidad total.

$$\Lambda = \frac{|W|}{|T|} = \prod_{j}^{\min(q-1,\,p)} \frac{1}{1+\lambda_j} \qquad \textbf{(3.11)}$$

La distribución de este estadístico es complicada por ello se han realizado diversas aproximaciones:

1.- Estadístico de Barlett:

$$H_0: \mu_1 = \mu_2 = ... = \mu_q$$

H_1: No todas las medias son iguales.

El estadístico de Barlett es una función de Λ y se aproxima a una χ^2.

$$V = -\left\{N-1-\frac{p+q}{2}\right\}\ln\Lambda \rightarrow \chi^2_{p(q-1)} \qquad \textbf{(3.12)}$$

Con este test se trata de ver si las *p* variables clasificadoras realmente contribuyen a discriminar entre las *q* clases. Si no se rechaza la hipótesis nula, no tendría sentido realizar el análisis. Esto es porque si las medias poblacionales de los grupos son iguales, las variables no tienen poder discriminante significativo.

2.- Descomposición del estadístico V.

$$\frac{1}{\Lambda} = \frac{|\boldsymbol{T}|}{|\boldsymbol{W}|} = |\boldsymbol{W}^{-1}||\boldsymbol{T}| = |\boldsymbol{W}^{-1}\boldsymbol{T}| = |\boldsymbol{W}^{-1}(\boldsymbol{W}+\boldsymbol{B})| = |\boldsymbol{I}+\boldsymbol{W}^{-1}\boldsymbol{B}| =$$

$$= (1+\lambda_1)(1+\lambda_2).....(1+\lambda_{\min(q-1,p)}) = \prod_{j}^{\min(q-1,p)} (1+\lambda_j) \qquad \textbf{(3.12)}$$

$$V = \left\{N-1-\frac{p+q}{2}\right\}\sum_{k=1}^{q-1}\ln(1+\lambda_q) \rightarrow \chi^2_{p(q-1)}$$

Si se rechaza la hipótesis nula se estaría aceptando que al menos un eje tiene poder discriminante. Además, se podría afirmar que el primer eje es estadísticamente significativo, ya que es que el mayor poder discriminante tiene. Si se ha rechazado la hipótesis de igualdad de medias, es decir, se acepta que el primer eje es significativo, se puede realizar el test sucesivamente para el resto de ejes. Para el segundo será:

$$V_1 = \left\{N-1-\frac{p+q}{2}\right\}\sum_{k=2}^{q-1}\ln(1+\lambda_q) \rightarrow \chi^2_{(p-1)(q-2)} \qquad \textbf{(3.13)}$$

y suponiendo que *q-1*< *p*, se tendrá de forma general

$$V_j = \left\{N-1-\frac{p+q}{2}\right\}\sum_{k=j+1}^{q-1}\ln(1+\lambda_q) \rightarrow \chi^2_{(p-j)(q-j-1)} \qquad j=0,1,2,..q-2 \qquad \textbf{(3.14)}$$

El proceso secuencial se detendrá cuando no se rechace la hipótesis nula. Algunas veces puede ocurrir que aunque se rechace la igualdad de medias, siendo $\boldsymbol{8}_i$ estadísticamente significativo, no interese retener tantos ejes por no considerarlos importantes en la práctica. Por tanto, es conveniente observar el valor de $\frac{\lambda_i}{\sum_j \lambda_j}$. Si este

valor es pequeño, puede no interesar la función discriminante asociada a ese autovalor aunque sea estadísticamente significativo.

b) Evaluación de la bondad de discriminación

Como medida de la bondad de ajuste se utiliza el coeficiente *eta cuadrado* (η^2).

$$\eta_j^2 = \frac{\lambda_j}{1+\lambda_j} \tag{3.15}$$

Esta medida es el coeficiente de determinación que se obtendría al realizar la regresión entre la variable categórica que indica la pertenencia a una clase y las puntuaciones discriminantes para la *j*-ésima función discriminante Es decir, se puede interpretar este coeficiente como el porcentaje de varianza de la función discriminante explicada por la diferencia entre grupos.

A la raíz cuadrada de este coeficiente se le denomina *correlación canónica*, toma valores entre 0 y 1 de forma que cuanto más se acerque a 1, mayor es la potencia discriminante de la *j*-ésima función discriminante.

$$\eta_j = \sqrt{\frac{\lambda_j}{1+\lambda_j}} \tag{3.16}$$

3.2.3. Interpretación de las funciones discriminantes y cálculo de la contribución de cada variable

Existe una relación cercana entre la interpretación de la función discriminante y la determinación de la contribución de cada variable a la separación entre las clases. Sin embargo, entre estas dos tareas hay una diferencia fundamental, y es que en el primer caso interesa el signo de los coeficientes de la función discriminante, mientras que en el segundo caso no. Para estudiar la contribución de cada variable a la discriminación entre grupos se proponen distintos métodos.

a) Coeficientes estandarizados

Para la *m*-ésima observación del grupo *k*-ésimo se puede expresar la función discriminante como:

$$d_{km} = (u_1^*)\frac{x_{1\,km}}{S_1} + (u_2^*)\frac{x_{2\,km}}{S_2} + \ldots\ldots + (u_p^*)\frac{x_{p\,km}}{S_p} \qquad \textbf{(3.17)}$$

donde S_j es la desviación estándar de la variable *j*-ésima dentro de cada grupo, es decir, la raíz cuadrada del elemento *j*-ésimo de la diagonal principal de la matriz $\mathbf{S_{pooled}} = \mathbf{W} / (N\text{-}q)$.

Por tanto, los coeficientes estandarizados tienen la forma: $u_j^* = S_j\, u_j \quad ; j = 1, 2,.., p$

En forma vectorial se tiene $\mathbf{U}^* = (\mathbf{diag\ S_{pooled}})^{1/2}\ \mathbf{U}$

Los coeficientes estandarizados reflejan la contribución de cada variable a la función o funciones discriminantes en presencia de las otras variables, ya que se han obtenido a partir de los autovalores de $\mathbf{W}^{-1}\ \mathbf{B}$, que tiene en cuenta la covarianza entre las variables. Estos coeficientes, en valor absoluto, pueden utilizarse para establecer un ranking en cuanto a la contribución de cada variable a la discriminación. Por otro lado, si se atiende también al signo de los coeficientes, se puede interpretar el significado de la función discriminante.

b) Criterio de las F`s parciales

Se calcula para cada variable clasificadora un test F parcial mostrando la separación debida a esta variable después de ajustar por el resto de variables clasificadoras. Según el valor de la F, las variables pueden ser ordenadas en cuanto a su significación. Para ello se calcula la lambda parcial

$$\Lambda(X_j / X_1, X_2, .., X_{j-1}, X_{j+1}, .., X_p) = \frac{\Lambda_p}{\Lambda_{p-1}}$$

$$F = \frac{1-\Lambda}{\Lambda}\frac{N-q-p+1}{q-1} \rightsquigarrow F_{q-1,\, N-q-p+1} \tag{3.18}$$

La diferencia entre los coeficientes estandarizados y los valores F parciales en el caso de varios grupos, está en:

- Con los coeficientes estandarizados se puede ver la contribución de cada variable a cada una de las funciones discriminantes, mientras que con los valores F se obtiene una idea de la contribución de cada variable a la separación general entre grupos, es decir, a la discriminación entre las clases teniendo en cuenta todas las dimensiones.

- En cuanto a la ordenación, la proporcionada por los valores F parciales coincidirá con la dada por los coeficientes estandarizados para la primera función discriminante, sobre todo si la importancia relativa de ésta es muy alta.

3.2.4. Selección de variables

Si se dispone de un número elevado de variables puede ocurrir que algunas de ellas sean irrelevantes en el sentido de discriminar entre los grupos. También puede ocurrir que la capacidad discriminante de alguna variable se deba casi exclusivamente a la correlación con otras variables clasificadoras.

En este caso, sería interesante poder realizar una selección de las variables con mayor capacidad discriminante. Para llevar a cabo esta selección se van a utilizar los valores de las F parciales. Cuanto mayor sea el valor de la F más significativa será la variable para la cual se ha calculado. Hay varios procedimientos:

a) Selección hacia delante (Forward)

En primer lugar hay que fijar un valor umbral para F, que se denomina valor *F de entrada*. Los pasos siguientes son:

1.- Calcular Λ para cada variable y su F asociada. Se elige la variable con menor Λ (o mayor F) Y X_1.

2.- Se calcula $\Lambda(X_j \backslash X_1)$ para cada una de las *p-1* variables que no entraron en el primer paso y se incluye en el análisis la variable con menor Λ o mayor F Y X_2.

3.- Se calcula $\Lambda(X_j \backslash X_1, X_2)$ y la F parcial asociada e incluimos en el análisis la variable con menor Λ o mayor F.

Es decir, en la primera etapa se incluye la variable que más discrimine individualmente y en sucesivas etapas se van incluyendo las variables que más discriminen en conjunto con las que ya han entrado. El proceso continúa hasta que ninguna variable alcance el valor mínimo o umbral F de entrada fijado.

b) Selección hacia atrás (Backward)

En principio se incluyen todas las variables y se fija un valor umbral de *F de salida*. En cada etapa sucesiva se va excluyendo del análisis la variable que menos discrimina hasta que ninguna variable presente un valor F parcial menor al prefijado.

c) Selección paso a paso (Stepwise)

Este método combina características de los dos métodos anteriores. En este caso, en cada paso pueden entrar y salir variables, por lo que es necesario fijar dos valores umbrales, el de entrada y el de salida.

Los pasos a seguir son:

1.- Se calculan las **7** y F univariantes y se selecciona la variable con menor **7** (o mayor F) Y X_1.

2.- Se calculan las **7** y F parciales para cada variable mediante

$$\Lambda(X_j / X_1) = \frac{\Lambda(X_j, X_1)}{\Lambda(X_1)} \qquad \textbf{(3.19)}$$

y se selecciona X_2. En este paso se debe comprobar si al haber incluido X_2 en el análisis X_1 sigue siendo significativa, si no lo es saldrá.

3.- Suponiendo que X_1 y X_2 han sido incluidas en el análisis. Se calculan

$$\Lambda(X_j / X_1, X_2) = \frac{\Lambda(X_j, X_1, X_2)}{\Lambda(X_1, X_2)} \qquad \textbf{(3.20)}$$

seleccionando X_3 siguiendo el mismo criterio de los pasos anteriores.

Una vez incluida X_3 , se comprueba si X_1 y X_2 deben seguir incluidas mediante **7**(X_1/ X_2, X_3) y **7**(X_2 / X_1, X_3).

El proceso termina cuando ninguna variable presente un valor F mayor que el establecido para entrar. Si el valor F de entrada es igual al de salida podría ocurrir que el proceso nunca llegara al fin. Para evitar este problema se determina un valor de F de salida menor que el valor de F de entrada.

d) El nivel de tolerancia

Es una medida de asociación lineal entre las variables clasificadoras. La tolerancia para la variable *j*-ésima es:

$$T_j = 1 - r_j^{\,2}$$

donde $r_j^{\,2}$ es el coeficiente de determinación de la regresión múltiple de la variable *j*-ésima sobre el resto de las variables clasificadoras seleccionadas.

Un nivel de tolerancia muy pequeño implica que esa variable está muy correlacionada con el resto de variables clasificadoras y, por tanto, aporta poca información novedosa. Se suele fijar como nivel mínimo aceptable el valor 0,001 para incluir una variable en el análisis.

3.2.5. Clasificación de los individuos

A la hora de clasificar un individuo existen diferentes procedimientos basados en la comparación de un caso con los centroides de grupo, a fin de determinar a cuál de ellos resulta más próximo. A continuación, se citan algunos de estos procedimientos.

a) Funciones de clasificación[3]

Este primer procedimiento se basa en las denominadas funciones de clasificación, que básicamente consisten en las funciones discriminantes calculadas para cada grupo. De esta forma, podemos definir la función de clasificación para el grupo *k* como:

$$F_k = a_{k0} + a_{k1} X_1 + a_{k2} X_2 + \ldots\ldots + a_{kp} X_p \qquad \textbf{(3.21)}$$

donde F_k es la función de clasificación para el grupo *k* y los coeficientes *a* se obtienen como[4]:

[3] Algunos autores las llaman “funciones discriminantes lineales de Fisher” y a las estudiadas previamente, “funciones canónicas discriminantes”.

[4] Ver Rencher (1995, pág 332)

$$a_{ki} = (N-q)\sum_{j=1}^{p} z_{ij}\,\bar{x}_{kj} \qquad \textbf{(3.22)}$$

siendo a_{ki} el coeficiente que multiplica a la variable *i* en el caso de la función de clasificación para el grupo *k*; $\bar{x}_{kj}$es la media de la variable *j* en el grupo *k* y z_{ij} es el elemento correspondiente de la matriz inversa de **W** (matriz de suma de cuadrados y productos cruzados intragrupos). La constante que se suma en la función de clasificación se determina de la siguiente manera:

$$a_{k0} = -0{,}5\sum_{j=1}^{p} a_{kj}\,\bar{x}_{kj} \qquad \textbf{(3.23)}$$

Estas puntuaciones F tienen la propiedad de que resultan más elevadas cuanto mayor sea la proximidad del caso al grupo. Por tanto, cuando haya que clasificar a un individuo se calcularán sus puntuaciones en cada una de las funciones de clasificación asignándolo a aquel grupo, o clase, en el que obtiene la puntuación F más alta.

Este procedimiento de clasificación resulta muy sensible a la violación del supuesto de igualdad de matrices de varianzas-covarianzas. Cuando no se verifica dicho supuesto, los casos tienden a ser clasificados en el grupo en el que se presente la mayor dispersión.

b) Cálculo de probabilidades de pertenencia a un grupo

El método anterior determina el grupo al que pertenece un individuo pero, en ocasiones, puede ser interesante conocer la probabilidad de su pertenencia a cada grupo, ya que ello permite realizar análisis más matizados e incluir otras informaciones, tales como la información *a priori* o los costes que implica un error en la clasificación.

El cálculo de las probabilidades se va a realizar utilizando el teorema de Bayes que permite el cálculo de las probabilidades *a posteriori* a partir de las probabilidades *a priori* y de la información muestral contenida en las puntuaciones discriminantes. Considerando, como hasta ahora, el caso general de *q* grupos, el teorema de Bayes establece que la probabilidad a posteriori de pertenencia a un grupo *k* con una puntuación discriminante *D* (*Prob(k/D)*) es la siguiente:

$$Prob(k / D) = \frac{\pi_k \, Prob(D / k)}{\sum_{j=1}^{q} \left(\pi_j \, Prob(D / j)\right)} \qquad \textbf{3.24)}$$

en el segundo miembro aparecen las probabilidades a priori ($\mathbf{B}$) y las condicionadas *Prob(D/k)* que se obtienen calculando la probabilidad de la puntuación observada suponiendo la pertenencia a un grupo *k*.

Si no se dispone de información a priori, se supondrá que la probabilidad a priori de todos los grupos o clases es la misma, es decir, $\mathbf{B}_1 = \mathbf{B}_2 = ... = \mathbf{B}_q = 1/q$, por lo que éstas no afectarán al cálculo de las probabilidades a posteriori.

Bajo las hipótesis de homoscedasticidad y normalidad, la probabilidad de pertenencia a cada grupo dada la puntuación discriminante obtenida se calcula como:

$$Prob(k / D) = \frac{e^{F_k}}{\sum_{j=1}^{q} e^{F_j}} \qquad k=1,2,...q \qquad \textbf{(3.25)}$$

donde F_k y F_j son las funciones definidas en la ecuación (3.20). Cada individuo se clasifica en el grupo para el que su probabilidad a posteriori sea mayor.

La ecuación anterior se puede modificar fácilmente en el caso en que las probabilidades a priori sean conocidas y quedaría como

$$Prob(k/D) = \frac{\pi_k e^{F_k}}{\sum_{j=1}^{q} \pi_j e^{F_j}} \qquad k=1,2,...q \tag{3.26}$$

interpretándose del mismo modo que la anterior.

Podría ocurrir que las probabilidades de que un sujeto pertenezca a dos clases distintas no sean muy diferentes entre sí, en cuyo caso, aún asignándolo a la clase en la que cuenta con mayor probabilidad, su clasificación no sería tan clara. Por ese motivo, resulta interesante conocer para cada individuo no sólo la máxima probabilidad, sino también las probabilidades de pertenecer a otros grupos.

c) Funciones de distancia generalizada

Por último, se puede citar un tercer procedimiento para la clasificación de un caso, basado en el cálculo de su distancia a los centroides de cada uno de los grupos. El caso sería asignado a la clase con cuyo centroide existe una menor distancia. Entre las distintas funciones de distancia que existen, la distancia de Mahalanobis resulta adecuada para valorar la cercanía entre casos y centroides. Para calcular la distancia cuadrada entre el punto *x* (correspondiente a un caso) y el centroide del grupo *k*, que se denota por C_k usaremos la expresión:

$$D^2(X/C_k) = (N-q)\sum_{j=1}^{p}\sum_{i=1}^{p} z_{ij}\,(\bar{x}_i - \bar{x}_{ik})\,(\bar{x}_j - \bar{x}_{jk}) \tag{3.27}$$

donde z_{ij} tiene el mismo significado que en el cálculo de los coeficientes de la función de clasificación. El empleo de esta distancia asume la igualdad de matrices de

covarianza para los grupos, condición que en caso de no cumplirse llevaría a tener que modificar la expresión anterior.

Un caso será clasificado en el grupo respecto al cual presenta la distancia más pequeña. Ello significa que el centroide de ese grupo es el que tiene un perfil más parecido al perfil del caso sobre las variables utilizadas para discriminar.

3.3. DISCRIMINANTE CUADRÁTICO

El discriminante cuadrático es similar al discriminante lineal, pero la frontera entre dos regiones distintas puede ser ahora una superficie cuadrática. Si se omite el supuesto de que las matrices de covarianzas son iguales, entonces en el razonamiento máximo verosímil con distribuciones normales se obtiene una superficie cuadrática (por ejemplo: elipse, hipérbole,..etc.) Este tipo de discriminación puede llevar a clasificaciones donde el conjunto de valores de los atributos para una clase esté muy próximo a los de otra. Clarke et al (1979) dicen que el procedimiento discriminante cuadrático es robusto para pequeñas distancias de la normalidad y que una fuerte curtosis (distribuciones con colas más grandes que la normal) no reduce sustancialmente la precisión. Sin embargo, el número de parámetros a estimar es $qp(p+1)/2$ y es necesario que las diferencias entre las varianzas sean considerables para justificar el uso de este método, sobre todo para conjuntos pequeños o de tamaño moderado (Marks y Dunn, 1974). Algunas veces las diferencias en las covarianzas se deben a la escala y es posible simplificarlas (Kendall et al, 1983). El discriminante lineal es todavía efectivo cuando no se separa mucho de la igualdad de covarianzas (Gilbert, 1969). Algunos aspectos de la dependencia cuadrática se pueden incluir en la forma lineal o logística (que se verá a continuación) adjuntando nuevos atributos que son funciones cuadráticas de los atributos dados.

3.3.1. La aplicación práctica del discriminante cuadrático

La función discriminante cuadrática se define más fácilmente como el logaritmo de la función de densidad apropiada, por tanto para cada clase se calcula un discriminante

cuadrático. El procedimiento usado toma el logaritmo de la función de densidad y sustituye las medias y matrices de covarianzas poblacionales por sus valores muestrales, obteniendo las llamadas estimaciones adaptadas (plug-in).Tomando el logaritmo de la ecuación de la normal multivariante y permitiendo distinguir las probabilidades a priori de las clases $\mathbf{B}_i$, se obtiene

$$\log \pi_i f_i(x) = \log \pi_i - \frac{1}{2}\log(|\Sigma_i|) - \frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) \quad \textbf{(3.28)}$$

como el discriminante cuadrático para la clase A_i. El subíndice i se refiere a la muestra de valores de clase A_i.

Para clasificar una nueva observación, se calcula el discriminante cuadrático para cada clase y se elige el mayor de ellos cuya clase será asignada. Para calcular las probabilidades a posteriori de las clases explícitamente, se toma la exponencial del discriminante y las cantidades resultantes se normalizan para que sumen uno (ver sección 2.3). Por tanto las probabilidades a posteriori de las clases $P(A_i/x)$ vienen dadas por:

$$P(A_i/x) = \exp[\log \pi_i - \frac{1}{2}\log(|\Sigma_i|) - \frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)] \quad \textbf{(3.29)}$$

sin tener en cuenta el factor de normalización.

Si hay una matriz de coste, entonces, sea cual sea el número de clases, el procedimiento más sencillo es calcular las probabilidades de las clases $P(A_i/x)$ y asociar los costes esperados explícitamente, utilizando las fórmulas de la sección 2.3. Un problema frecuente en el discriminante cuadrático es que un atributo tenga varianza cero en una clase, porque entonces la matriz de covarianzas no es invertible. Una posible solución es añadir una pequeña constante positiva a los valores de la diagonal de la matriz (esto se corresponde con añadir ruido aleatorio a los atributos). Otra forma, es utilizar alguna combinación de la covarianza de la clase y la covarianza conjunta. Al igual que en el apartado anterior sustituyendo los valores poblacionales por los

muestrales correspondientes, se obtiene la versión adaptada de estas fórmulas. Muchos paquetes estadísticos permiten trabajar con el discriminante cuadrático (por ejemplo MINITAB, SAS.)

3.3.2. Regularización y estimaciones alisadas

El principal problema del discriminante cuadrático es el gran número de parámetros a estimar y la varianza tan grande que resulta de los discriminantes estimados. Un problema citado es la existencia de valores iguales o cercanos a cero en la matriz de covarianzas muestrales. Los intentos de aliviar este problema se conocen como métodos de regularización y el de mayor utilidad en la práctica fue desarrollado por Friedman (1989), quien propuso un compromiso entre el discriminante lineal y el cuadrático, a través de una familia con dos parámetros de estimación. Un parámetro controla el suavizado de la matriz de covarianzas estimada de la clase, mientras que el segundo parámetro evita que la matriz de covarianzas sea singular.

El suavizado estimado de la matriz de covarianzas de la clase *i* es

$$(1-\delta_i)S_i + \delta_i S \qquad \textbf{(3.30)}$$

donde S_i es la matriz de covarianzas muestral de la clase *i* y *S* es la matriz de covarianzas conjunta. Cuando δ_i es cero no hay suavizado y la matriz de covarianzas estimada para la clase *i* coincide con la matriz de covarianzas muestral *i*-ésima, S_i. Cuando $\delta_i = 1$, todas las clases tienen la misma matriz de covarianzas, llamada matriz de covarianzas conjunta (en inglés *pooled*). Friedman (1989) hace el valor de δ_i más pequeño para las clases con mayor número de observaciones. Para la muestra *i*-ésima con n_i observaciones:

$$\delta_i = \frac{\delta(N-q)}{\delta(N-q)+(1-\delta)(n_i-1)} \qquad \textbf{(3.31)}$$

donde $N = n_1 + n_2 + n_3 .... + n_q$.

El otro parámetro, **8**, es un pequeño término constante que es añadido a las diagonales de las matrices de covarianzas, esto se hace para que la matriz de covarianzas no sea singular y al mismo tiempo suaviza las matrices de covarianzas. Como se mencionó en el discriminante lineal, cualquier singularidad de la matriz de covarianzas causará problemas y, como en discriminante cuadrático hay una matriz de covarianzas para cada clase, la probabilidad de un problema de este tipo es mayor, sobre todo para las clases con tamaños muestrales pequeños.

Esta familia de procedimientos con dos parámetros la describe Friedman como "análisis discriminante regularizado". Varios procedimientos sencillos se incluyen como casos particulares: discriminante lineal ordinario (*****=1 ; **8**=0); discriminante cuadrático (*****=0 ; **8**=0) y los valores *****=1 ; **8**=1 corresponden a una regla de mínima distancia euclídea.

En la práctica, el usuario tiene que decidir, por ensayo error, elegir los valores ***** y **8**. Friedman (1989) da varios métodos de atajo para reducir la cuantía de los cálculos.

3.4. DISCRIMINANTE LOGÍSTICO

Al igual que el discriminante lineal, la regresión logística opera eligiendo un hiperplano para separar las clases lo mejor posible, pero cambia el criterio para una buena separación. El discriminante lineal de Fisher optimiza una función de coste cuadrática mientras que en el discriminante logístico se maximiza una probabilidad condicionada. Sin embargo, en la práctica, frecuentemente hay una diferencia pequeña entre las dos y el discriminante lineal proporciona valores de partida buenos para el logístico. El discriminante logístico es idéntico, en teoría, al discriminante lineal para distribuciones normales con covarianzas iguales y también para atributos independientes dos a dos, por lo que las mayores diferencias entre los dos son de esperar cuando se esté lejos de esos dos casos, por ejemplo cuando los atributos tengan distribuciones muy diferentes de la Normal con covarianzas muy distintas.

El método es sólo parcialmente paramétrico, puesto que las verdaderas funciones de densidad para las clases no están modelizadas, pero sí los ratios entre ellas. En concreto, los logaritmos de $\mathbf{B}_1$ /$\mathbf{B}_2$ veces los ratios de las funciones de densidad para las clases se modelizan como funciones lineales de los atributos. Así, para dos clases

$$\log \frac{\pi_1 f_1(x)}{\pi_2 f_2(x)} = \alpha + \beta' x \qquad \textbf{(3.32)}$$

donde " y el vector p-dimensional $ son los parámetros del modelo que deben ser estimados. El caso de distribuciones normales con covarianzas iguales es un caso particular, para el cual los parámetros son funciones de las probabilidades a priori, las medias de las clases y la matriz de covarianzas común. Sin embargo, el modelo también cubre estos casos, como cuando los atributos son independientes con valores 0 ó 1. Uno de los atractivos es que la escala discriminante cubre todos los números reales. Un valor grande positivo indica que la clase A_1 es más probable, mientras que un valor grande negativo indica que la clase A_2 es más probable.

En la práctica los parámetros se estiman por máxima verosimilitud condicionada. El modelo implica que, dados los valores que toma x en los atributos, las probabilidades condicionadas para las clases A_1 y A_2 toman la forma:

$$P(A_1/x) = \frac{\exp(\alpha + \beta' x)}{1 + \exp(\alpha + \beta' x)} \qquad \textbf{(3.33)}$$

$$P(A_2/x) = \frac{1}{1 + \exp(\alpha + \beta' x)} \qquad \textbf{(3.34)}$$

respectivamente. Si se tienen muestras independientes de las dos clases, la probabilidad condicionada de los parámetros " y $ será:

$$L(\alpha, \beta) = \prod_{\{muestra\ A_1\}} P(A_1/x) \prod_{\{muestra\ A_2\}} P(A_2/x) \qquad \textbf{(3.35)}$$

y los parámetros estimados son los valores que maximizan esta probabilidad. Se calculan por métodos iterativos, como propusieron Cox (1966) y Day y Kerridge (1967). Los modelos logísticos pertenecen a la clase de Modelos Lineales Generalizados (GLMs, generalized linear models), que generalizan el uso de modelos de regresión lineal para tratar variables aleatoreas no normales y en particular con variables binomiales. En este contexto, la variable binomial es un indicador que informa si un ejemplo es de la clase A_1 o no. Cuando hay más de dos clases, se toma una clase de referencia y hay *q-1* conjuntos de parámetros para las apuestas de cada clase en relación a la clase de referencia. Para exponer este caso, se abrevia $\alpha + \beta' x$ por $\beta' x$, en esta sección de aquí en adelante, x es un vector $(p+1)$-dimensional cuyo primer término es la unidad y el primer término en $ corresponde a la constante ".

De nuevo, los parámetros se estiman por máxima verosimilitud condicionada. Dados los valores *x* en los atributos, la probabilidad condicionada de la clase A_i , donde $i..q$ y la probabilidad condicionada de la clase A_q , son:

$$P(A_i/x) = \frac{\exp(\beta_i' x)}{\sum_{j=1,2,...q} \exp(\beta_j' x)} \qquad \textbf{(3.36)}$$

$$P(A_q/x) = \frac{1}{\sum_{j=1,2..q} \exp(\beta_j' x)} \qquad \textbf{(3.37)}$$

Si se tienen muestras independientes de las *q* clases, las probabilidades condicionadas de los parámetros $$_i$ son:

$$L(\beta_1, \beta_2, ..., \beta_{q-1}) = \prod_{\{muestra\ A_1\}} P(A_1/x) \prod_{\{muestra\ A_2\}} P(A_2/x).... \\ ..... \prod_{\{muestra\ A_q\}} P(A_q/x) \qquad \textbf{(3.38)}$$

Nuevamente la estimación de los parámetros son aquellos valores que maximizan esta probabilidad.

En la forma básica del método, un ejemplo se asigna a la clase para la cual la probabilidad a posteriori es mayor si es positiva, o a la clase de referencia si todas las probabilidades a posteriori son negativas.

Se pueden componer modelos más complicados añadiendo transformaciones de los atributos dados, por ejemplo: productos o pares de atributos. Como se mencionó en la sección 3.1, cuando hay atributos categóricos con r posibles valores, siendo $r > 2$, generalmente será necesario convertirlos en $r-1$ atributos binarios antes de utilizar el algoritmo, sobre todo si las categorías no están ordenadas. Anderson (1984) señala que puede ser apropiado incluir transformaciones o productos de los atributos en la función lineal, pero que para grandes conjuntos esto supone demasiados cálculos. Una forma de aumentar la complejidad del modelo, sin perder inteligibilidad, es añadir parámetros, pudiéndose establecer vínculos con modelos gráficos y árboles múltiples (Polytrees).

3.4.1. La aplicación práctica del discriminante logístico

La mayoría de paquetes estadísticos utilizan el análisis discriminante lineal para dos clases. SYSTAT tiene además una versión de regresión logística capaz de manejar problemas con más de dos clases. Si un paquete tiene sólo regresión logística binaria (dos clases), Begg y Gray (1984) sugieren un procedimiento aproximado mediante el cual todas las clases se comparan con una de referencia por regresión logística y los resultados se combinan después. La aproximación obtiene en la práctica un buen resultado según estos autores.

Muchos paquetes estadísticos (GLIM, Splus, Genstat) incluyen una función del GLM (Generalised Linear Model), posibilitando una fácil programación de la regresión logística, en un código de dos o tres líneas. El procedimiento es definir una variable indicador para los sucesos de clase A_1 . La variable indicador es una variable binomial

que tiene como valores a la función vínculo logit y a la regresión generalizada. Para ese propósito se utiliza el paquete Splus. Esto funciona bien para dos clases y tiene el mérito de requerir un esfuerzo extra de programación pequeño. Para más de dos clases, la complejidad del problema aumenta sustancialmente y, aunque técnicamente es posible utilizar los procedimientos GLM, el esfuerzo de programación es mucho mayor y menos eficiente.

La solución máximo verosímil puede hallarse por un procedimiento iterativo de Newton-Raphson y es fácil obtener las derivadas necesarias de la verosimilitud (o, equivalentemente, el logaritmo de la verosimilitud). El procedimiento inicial más sencillo es igualar los coeficientes $\$_i$ a cero excepto para los primeros coeficientes $\"_i$ que se igualan a los logaritmos del número de ejemplos en cada clase, por ejemplo: $\"_i = \log n_i$, donde n_i es el número de ejemplos de clase A_i. Esto asegura que los valores $\$_i$ son aquellos del discriminante lineal después de la primera iteración.

Una alternativa sería usar los parámetros del discriminante lineal como valores de partida. En las iteraciones siguientes, aunque a veces hay que reducir el número de pasos, el procedimiento normalmente converge en aproximadamente diez iteraciones. Sin embargo, cada iteración requiere un nuevo cálculo del Hessiano y es esto lo que requiere la mayor parte del trabajo computacional. El Hessiano es una matriz cuadrada con *(q-1)(p+1)* filas y cada término requiere un sumatorio sobre todas las observaciones en el conjunto de datos (aunque gracias a la simetría del Hessiano se reduce el cálculo necesario). A pesar de esto hay del orden de $q^2 p^2 N$ cálculos necesarios para obtener el Hessiano en cada iteración. En un ejemplo con $q = 10$; $p = 40$ y $N = 9000$, el número de operaciones es de 10^9 en cada iteración. En tales casos, es preferible utilizar un procedimiento de búsqueda puramente numérico.

Otra solución cuando el procedimiento de Newton-Raphson requiere demasiado tiempo, es utilizar un método basado en un Hessiano aproximado. La aproximación utiliza el hecho de que el Hesssiano para la iteración de orden cero es simplemente una réplica de la matriz de covarianzas utilizada por la regla discriminante lineal. Este

Hessiano es utilizado para todas las iteraciones. En situaciones donde hay poca diferencia entre los parámetros logísticos y lineales, la aproximación es buena y converge rápidamente (aunque generalmente se requiere alguna iteración más). Sin embargo, cuando más interesa, es decir, cuando los parámetros logísticos y lineales son muy distintos, la convergencia por este método es muy lenta, e incluso después de cien iteraciones puede estar lejos de converger. Lo normal es parar después de cincuenta iteraciones, aunque los valores de los parámetros generalmente no están estabilizados, las clases predichas para los valores están razonablemente estables, por tanto la potencia predictiva de la regla resultante no debe verse afectada seriamente.

3.5. MÉTODO DEL VECINO MÁS PRÓXIMO

Las reglas de clasificación por vecindad se basan en la búsqueda en un conjunto de datos de los *k* ejemplos más cercanos al ejemplo a clasificar. El conjunto de datos en el que se lleva a cabo la búsqueda de los *k* vecinos más próximos puede ser el conjunto original de datos, *T*, o bien un subconjunto de éste, por lo que en general se le llama conjunto de referencia (*R*) o de aprendizaje. Precisamente esta búsqueda es el principal inconveniente de estos métodos, ya que el tiempo empleado en cada nueva clasificación es elevado, por ello han surgido diversos criterios, algunos de los cuales se comentarán más adelante y que, en general, intentan reducir el conjunto de referencia, como por ejemplo los métodos de edición y de condensado.

Para poder determinar la cercanía entre la nueva observación y los ejemplos del conjunto de aprendizaje se debe especificar la métrica a utilizar. En Wilson y Martínez (2000) se pueden encontrar diversas medidas, entre ellas la más extendida es la distancia Euclídea (E), que se define como:

$$E(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2} \qquad \textbf{(3.39)}$$

donde x e y son los dos vectores de datos cuya distancia se quiere calcular, p es el número de atributos y x_i e y_i son los valores del atributo i-ésimo en ambas observaciones. Esta función es apropiada cuando todos los atributos son numéricos y tienen rangos de aproximadamente la misma amplitud. Cuando estas variables tienen rangos significativamente diferentes, pueden ser normalizadas dividiendo las distancias individuales de los atributos por el rango o desviación estándar de los mismos, calculando así lo que se conoce como distancia Euclídea Normalizada (EN), que matemáticamente se define como:

$$EN(x,y) = \sqrt{\sum_{i=1}^{p}\left(\frac{|x_i - y_i|}{\max_i - \min_i}\right)^2} \tag{3.40}$$

siendo max_i y min_i el mayor y menor valor, respectivamente, observados en el conjunto de entrenamiento para el atributo i-ésimo.

3.5.1. La regla 1-NN

El método más sencillo es la regla de clasificación del vecino más próximo (1-NN). En este caso a la nueva observación se le asigna la clase correspondiente al ejemplo más cercano en R, que se puede representar por x_{NN}. Suponiendo que el número de ejemplos en el conjunto de referencia es N, esta regla asigna la clase de aquel ejemplo cuya distancia a la nueva observación (E) sea la más pequeña. Esto se puede expresar como

$$asignar\ A_d\ si \begin{cases} E(x,x_{NN}) = \min\limits_{i=1,2,\ldots N}\{E(x,x_i)\} \\ (x_{NN},A_d) \in R \end{cases} \tag{3.41}$$

Esta regla supone la división del espacio de representación en N *regiones de influencia*, una por cada ejemplo. Estas regiones tienen forma poligonal y están delimitadas por los puntos situados a igual distancia entre un ejemplo y su vecino más

próximo. Estas zonas se conocen como regiones de Voronoi y la partición poligonal del espacio de representación, como partición de Voronoi (el módulo espacial de S-plus calcula estas regiones).

Cada región de Voronoi constituye una región de decisión (respecto al ejemplo central), por lo que se puede averiguar la región de decisión de una clase mediante la unión de todas las regiones de Voronoi de los ejemplos de esa clase. Por tanto, los límites de decisión serán fronteras lineales a trozos.

3.5.2. La regla *k*-NN

Si se generaliza a los *k* vecinos más cercanos, se obtiene la regla de clasificación de los *k* vecinos más próximos (*k*-NN). Esta regla se basa en la suposición de que los ejemplos de la misma clase están cercanos entre sí, de tal forma que primero se identifican los *k* vecinos más próximos de la nueva observación que se quiere clasificar y, después, se le asigna la clase mayoritaria de entre esas *k* observaciones más cercanas.

El valor *k* es una constante, normalmente impar para evitar que se puedan producir empates, los valores más utilizados son 1, 3 y 5. Si se denota por $K_i\,(x)$ al número de ejemplos de clase A_i entre los *k* más próximos a *x*, esta regla puede expresarse como

$$asignar\ A_d\ si\ K_d(x) = \max_{i=1,2,\ldots q} \{K_i\,(x)\} \qquad \textbf{(3.42)}$$

3.6. COTAS DE ERROR DE LAS REGLAS 1-NN Y *K*-NN

Teóricamente se supone que el conjunto de entrenamiento es grande (N64) y aunque en la práctica este supuesto es poco realista, resulta interesante estudiar la tasa de error en el caso de que el conjunto de ejemplos sea infinito.

Si se denota por E* al error cometido por la regla óptima de Bayes y por E_1, al error cometido por la regla 1-NN, puede demostrarse (ver Devijver y Kittler (1982)) que

$$E^* \leq E_1 \leq E^*\left(2 - \frac{q}{(q-1)}E^*\right) \leq 2E^* \qquad \textbf{(3.43)}$$

Es decir, el error cometido por la regla 1-NN está acotado inferiormente por E* y superiormente, por aproximación, por dos veces E*.

De igual forma puede demostrarse que para la regla *k*-NN con un conjunto infinito de observaciones y con *k* suficientemente grande el error de la regla *k*-NN tiende a la tasa de error de Bayes.

$$\lim_{k\to\infty} E_k = E^* \qquad \textbf{(3.44)}$$

Es decir, cuanto mayor sea el número de vecinos tenidos en cuenta menor será el error cometido en la clasificación *k*-NN. Pero este resultado no es aplicable en la práctica por varios motivos, el primero es que el conjunto de aprendizaje nunca es infinito y, además, en muchas aplicaciones puede ser demasiado reducido. También hay que tener en cuenta la posibilidad de que existan errores en el conjunto de entrenamiento (observaciones mal clasificadas) que pueden conducir a errores en la clasificación de las nuevas observaciones. Además, la elección de *k* muy grande dará problemas a la hora de clasificar las observaciones situadas cerca de las fronteras de las reglas de decisión.

3.7. CLASES DE RECHAZO

Las reglas de clasificación de los k-NN pueden ampliarse para establecer un mínimo de ejemplos de la clase a asignar, es decir, si por ejemplo se consideran los cinco vecinos más próximos se podría exigir que al menos existan tres ejemplos de una clase A_d para asignarla a la nueva observación, en caso contrario se asignaría una clase A_0 que sería la clase por defecto.

Esto puede ser aconsejable en el caso de que exista una clase mayoritaria A_0, con una probabilidad a priori de ocurrir superior a las demás y, por tanto, salvo que exista una fuerte evidencia (mayoría cualificada) se puede asignar esa clase, con un riesgo pequeño.

En este caso se habla de la regla de (*k*, *t*)-NN:

$$asignar \begin{cases} A_d \; si \; K_d(x) = \max\limits_{i=1,2,...q} \{K_i(x)\} \geq t \\ A_0 \; en \; otro \; caso \end{cases} \tag{3.45}$$

Por tanto, se asigna la clase A_d, si además de ser la de mayor presencia entre los *k* vecinos más cercanos, tiene un número de representantes superior al umbral *t*, donde $t < k$.

Cuando la importancia de las clases no es similar, se pueden establecer distintos umbrales para cada clase, lo que permite un mayor control sobre el grado de confianza exigido para asignar la clase A_d. Esta regla se conoce como la regla (k, t_d)-NN:

$$asignar \begin{cases} A_d \; si \; K_d(x) = \max\limits_{i=1,2,...q} \{K_i(x)\} \geq t_d \\ A_0 \; en \; otro \; caso \end{cases} \tag{3.46}$$

Es decir, en este caso se tiene un umbral para cada clase t_d, de tal forma que a aquellas clases cuya pertenencia suponga un mayor coste o riesgo se les exigirá una mayor seguridad que a aquellas cuya pertenencia no suponga coste o riesgo alguno. Por ejemplo, si hubiese que decidir si un acusado es culpable o inocente, primará la presunción de inocencia y hará falta probar, sin que quepa ninguna duda razonable, la culpabilidad para declararlo culpable, en este ejemplo sólo habría dos clases (culpable o inocente), pero es fácilmente extensible a más de dos clases.

En el caso de la regla 1-NN el umbral habrá que establecerlo en función de la distancia. La regla 1-NN (*t*) quedaría como:

$$asignar \begin{cases} A_d \, si \begin{cases} E(x, x_{NN}) = \min\limits_{i=1,2,\ldots N} \{E(x, x_i)\} \le t \\ (x_{NN}, A_d) \in R \end{cases} \\ \\ A_0 \, en \, otro \, caso \end{cases} \quad \textbf{(3.47)}$$

En este caso, la inclusión del umbral significa que se supone que las observaciones de la misma clase están, como mucho, a una distancia *t* y, superado ese umbral, la probabilidad de que sean de la misma clase se hace muy pequeña. La selección del valor *t* debe hacerse en función de la distribución de las distancias entre cada ejemplo y su vecino más próximo, es decir, se coge cada ejemplo del conjunto de entrenamiento y se calcula la distancia de éste a su vecino más próximo, y así para todos los ejemplos del conjunto de aprendizaje. La elección del valor exacto de *t*, dependerá de esa distribución de las distancias y también del objetivo buscado, por lo que, en la práctica, se recomienda seleccionarlo tras probar con varios valores de *t*.

3.8. COMENTARIOS SOBRE LAS REGLAS *K*-NN

En la práctica, la aplicación de las reglas de clasificación por vecindad presenta una serie de limitaciones, la primera, como ya se ha mencionado, se debe a la escasez de conjuntos suficientemente grandes por lo que las tasas de error, tanto de las reglas 1-NN como de las reglas *k*-NN, no se aproximan a sus límites respectivos 2E* y E* (siendo E*, la tasa de error de Bayes).

Además, por la propia naturaleza de estos procedimientos, el tiempo empleado en cada clasificación puede llegar a ser relativamente elevado ya que será necesario averiguar los *k* vecinos más próximos (en el caso general) cada vez que se presente una observación nueva, es decir, hay que calcular las distancias de esa nueva observación

respecto a todos los ejemplos del conjunto de entrenamiento y, después, ordenarlos de menor a mayor y elegir los *k* ejemplos con menor distancia. Esto hace que el coste computacional a la hora de clasificar sea elevado, aunque ciertos programas, como el S-Plus, realizan esta labor de forma casi inmediata.

Sea cual sea el valor de *k*, la búsqueda se lleva a cabo a lo largo de todo el conjunto de referencia, lo que significa que el coste de la búsqueda depende linealmente de N. Si se considera el coste de calcular cada distancia, dependerá, en el caso de la distancia euclídea, de la dimensionalidad de las observaciones, es decir, del número de atributos "*p*" que describen cada ejemplo. Por tanto, el coste global dependerá linealmente de *N* y *p*. Además, hay que considerar el espacio de almacenamiento requerido, que también es de orden O(*Np*).

Por tanto, cuando el conjunto de referencia sea considerablemente grande (lo que es deseable en términos de optimalidad), la aplicación de las reglas *k*-NN, tal como se han planteado, resulta casi imposible si, además de un conjunto de referencia numeroso, los datos son de alta dimensionalidad. En consecuencia, son dos los factores críticos para el coste computacional: el número de atributos que describe cada ejemplo (*p*) y el tamaño del conjunto de aprendizaje (*N*).

Para el primer problema se puede utilizar alguna de la técnicas de selección de características vistas en el apartado 2.5. Además, hay que tener en cuenta que la correcta selección de estas variables es muy importante, porque los métodos de clasificación basados en la vecindad generalmente se comportan muy bien cuando los atributos utilizados tienen buenas propiedades para la predicción. Pero con atributos pobres o redundantes, el comportamiento se verá seriamente perjudicado.

En cuanto al problema que presenta el tamaño del conjunto de referencia, existen estrategias para reducir el coste computacional de aplicar las reglas *k*-NN a todo el conjunto de entrenamiento. Se pueden agrupar en dos grandes grupos, según el procedimiento utilizado para conseguir el objetivo de reducir el coste en tiempo.

1. Reducir el conjunto de referencia.

Se busca un subconjunto del conjunto de entrenamiento que tenga las mismas propiedades que el conjunto original, para aplicar la regla de clasificación por vecindad sobre este subconjunto, con lo que el número de ejemplos para los que hay que calcular las distancias es menor. Aunque existe el riesgo de que el subconjunto no sea representativo del conjunto original o de la población en general y, por tanto, se pierda precisión a la hora de clasificar.

Entre estos métodos se encuentran los métodos de edición, orientados fundamentalmente, a aumentar la tasa de acierto del clasificador 1-NN eliminando ejemplos ruidosos y, por tanto, de forma indirecta reducen el tamaño del conjunto de entrenamiento. La reducción, aún siendo significativa, en muchas ocasiones puede no ser suficiente para conseguir el objetivo buscado de reducir el coste computacional. Métodos específicos de construcción de un conjunto de entrenamiento reducido y representativo son los métodos de condensado y los métodos de aprendizaje adaptativo.

2. Mejorar la eficiencia computacional de la búsqueda del vecino más próximo.

Estos métodos no reducen el conjunto de entrenamiento, sino que se centran en eliminar aquellos cálculos innecesarios consiguiendo reducir el número de operaciones. La utilización de estos métodos no representa ningún peligro de pérdida de generalidad, ya que únicamente se reduce el número de cálculos.

Entre estos métodos están los basados en ordenaciones, que organizan los ejemplos del conjunto de entrenamiento según alguna coordenada y la búsqueda la llevan a cabo de acuerdo a alguna propiedad de la métrica asociada al espacio.

También forman parte de estos métodos, los denominados métodos jerárquicos, que realizan una descomposición jerárquica del conjunto de entrenamiento, lo que implica

la organización de los ejemplos en una estructura de árbol para después aplicar una técnica de poda y ramificación (branch and bound) para explorar el árbol.

Ambos métodos, tanto los que persiguen la reducción del conjunto de referencia como los que buscan mejorar la eficiencia computacional, suponen reducir el coste de la búsqueda a cambio de un *coste de preprocesamiento*, que es el coste de seleccionar el conjunto reducido o construir el árbol de búsqueda. Por tanto, se debe tener en cuenta la existencia de ambos costes y, en cada aplicación, se ha de llegar a un equilibrio entre los dos, en función de la magnitud de ambos habrá que decidir si el coste de preprocesamiento debe ser asumido en aras a conseguir una reducción en el coste de búsqueda. También hay que valorar que, mientras el preprocesamiento sólo se realiza una vez, la búsqueda hay que realizarla cada vez que se tenga que clasificar una nueva observación.

3.9. REDUCCIÓN EN LAS REGLAS DEL VECINO MÁS PRÓXIMO

Existen diversos métodos para reducir el conjunto de entrenamiento, en Wilson y Martínez (2000) se puede encontrar una relación de varios de estos métodos, exponiendo sus principales diferencias. La mayoría de los algoritmos expuestos a continuación buscan un subconjunto S del conjunto original T, que reduzca el coste de almacenamiento, consiguiendo también en la mayoría de los casos mejorar la precisión. Cuando no se especifique lo contrario se utilizará k=1, aunque puede modificarse para k mayor que uno.

3.9.1. Regla del vecino más próximo condensado

Este método fue propuesto por primera vez por Hart en 1968 y su objetivo es incrementar la eficiencia computacional de la clasificación por vecindad mediante la selección de un subconjunto reducido y representativo del conjunto inicial de ejemplos a cambio puede producirse una pequeña reducción en la precisión de la clasificación. Este método se conoce en inglés somo Condensed Nearest Neighbour (CNN)

Este algoritmo busca un subconjunto *S* del conjunto *T*, tal que cada miembro de *T* está más cerca de un miembro de *S* de la misma clase, que de un miembro de *S* de clase diferente, suponiendo que no existen dos ejemplos en *T* que tengan los mismos valores en los atributos, pero clases diferentes. Como se acaba de decir, el método de condensado de Hart se basa en la propiedad de consistencia que dice lo siguiente: *Sc es consistente respecto a otro conjunto S, donde Sc ƒ S, si S puede clasificarse correctamente usando los elementos Sc como referencia.*

Si el conjunto *S* es representativo del problema, también lo será *S*c y, por tanto, el clasificador construido a partir de *S*c tenderá al clasificador construido a partir de *S*. Hart define un conjunto condensado como un conjunto consistente y reducido de ejemplos, por tanto no impone el requisito de que este subconjunto sea minimal. La exigencia de que el conjunto *S*c sea consistente respecto a *S* implica que los ejemplos de *S* se clasifican correctamente usando la regla 1- NN a partir del conjunto *S*c.

Este algoritmo es muy sensible al ruido, porque los ejemplos ruidosos serán clasificados incorrectamente por sus vecinos y, por tanto, pasarán a formar parte de *S*. Esto causa dos problemas:

En primer lugar, la reducción del conjunto de entrenamiento se verá afectada porque, al mismo tiempo que se añaden los ejemplos ruidosos a *S*, habrá que añadir también los ejemplos buenos para poderlos clasificar correctamente, porque si no se les asignaría la clase del ejemplo ruidoso.

También se ve prejudicada la precisión de la clasificación, ya que los ejemplos ruidosos que se incluyan en *S* se utilizarán a la hora de clasificar a las nuevas observaciones. Puede ocurrir que algún ejemplo ruidoso esté entre los *k* vecinos más próximos, o aún peor en el caso de 1-NN, y por tanto pueda llevar a errores en la clasificación. Además, como parte de los ejemplos no ruidosos han sido eliminados, el área de influencia de los ejemplos ruidosos es mayor que en *T*, pudiendo llegar al caso

de que la precisión de la clasificación sea peor que antes de aplicar el CNN. De todas formas este caso es poco probable y muy pesimista.

Por todo lo anterior, el conjunto a condensar suele ser previamente editado como se verá en el apartado 3.9.4. Es decir, en primer lugar el conjunto *S* de partida es editado, obteniendo el conjunto editado S_E, este conjunto es a su vez condensado utilizando el algoritmo de Hart, obteniéndose el subconjunto S_{EC}, que servirá de punto de referencia para que la regla de clasificación 1-NN asigne la clase correspondiente a la nueva observación *x*.

El algoritmo de condensado de Hart se basa en la selección de ejemplos que determinen las fronteras de decisión entre clases. El algoritmo es de tipo incremental y sin vuelta atrás, de tal forma que al conjunto condensado se le van añadiendo ejemplos que no pueden ser eliminados posteriormente. La frontera de decisión inducida por el clasificador 1- NN está determinada por los ejemplos más cercanos a esa frontera, por lo que interesa seleccionar y añadir los ejemplos situados cerca de las fronteras (eliminando o descartando los que están dentro de los agrupamientos). Para ello, se supone que aquellos ejemplos que sean correctamente clasificados por su vecino más próximo estarán en el interior de los agrupamientos, mientras que los ejemplos fronterizos tenderán a ser clasificados incorrectamente, siendo éstos los que interesa retener.

En concreto, el algoritmo de Hart se puede esquematizar de la siguiente manera:

1. Se inicia con *S*c vacío y $S_{aux} = S$.

2. Se van seleccionando de forma aleatoria, uno a uno, los ejemplos de *S*c, el primero de ellos siempre se añade a *S*c y se elimina por tanto de S_{aux}.

3. El segundo y sucesivos se clasifican con ejemplos retenidos en *S*c y, si esta clasificación es correcta, se deja en S_{aux} mientras que, si la clasificación es incorrecta se pasa de S_{aux} a *S*c.

4. Así se van presentando los ejemplos de S_{aux} hasta que en una iteración completa ningún ejemplo sea añadido a *S*c, es decir, todos los ejemplos de S_{aux} son clasificados correctamente por *S*c.

5. La segunda condición de parada será cuando S_{aux} este vacío o sea *S*c= S_{aux}. En la práctica es más frecuente la primera, ya que la segunda supone el fracaso del proceso de condensado, porque no se habría podido reducir el conjunto de partida.

3.9.2. Regla del vecino más próximo selectiva

Este método fue propuesto por Ritter en 1975 y es una extensión del CNN de Hart, que intenta conseguir que el subconjunto *S* sea mínimo. Para ello exige que cada miembro de *T* debe estar más cercano a un miembro de *S* de la misma clase, que a cualquier miembro de *T* (en lugar de *S*) de una clase distinta. El CNN, en cambio, exigía que estuviese más cerca de un miembro de *S* de la misma clase que a cualquier miembro de *S* de distinta clase.

El algoritmo SNN (Selective Nearest Neighbour Rule) es más complejo que la mayoría de los algoritmos de reducción y el tiempo de aprendizaje es significativamente mayor, debido a la utilización de una matriz binaria (0, 1) de orden *n* x *n*, que se llamará *A*, donde *n* es el número de ejemplos en *T*. Cada elemento de la matriz se representa por $A_{i\,j}$, que tomará el valor 1 cuando el ejemplo *j* es de la misma clase que el ejemplo *i*-ésimo, y está más cerca del ejemplo *i*-ésimo que el ejemplo más cercano de *i* en *T* de diferente clase. $A_{i\,i}$ es siempre igual a 1. $A_{i\,j}$ es cero en los demás casos.

Una vez que esa matriz está construida, se realizan los cinco pasos siguientes hasta que no quede ninguna columna en la matriz.

1. Se buscan todas las columnas *j* que tengan solamente un 1 y, suponiendo que la fila donde la columna *j*-ésima tiene un 1 es la fila *i*, todas las columnas que tengan un 1 en la fila *i* son eliminadas y también la fila *i*. El ejemplo *i* se añade a *S*.

2. Se buscan aquellas filas *i* que, para todas las columnas *j* (de las que quedan) y para alguna fila *k* (de las que quedan), cumplan $A_{ij} \# A_{kj}$ y se elimina la fila *i*. En otras palabras, la fila *i* se elimina si para otra fila *k*, siempre que la fila *i* contenga un 1, la fila *k* también lo contiene. En este caso el ejemplo *i* no se añade a *S*.

3. Se buscan todas las columnas *j* tales que, para todas las filas *i* y alguna columna *k* (de las queden en ambos casos), se cumpla $A_{ij} \$ A_{ik}$ y se elimina la columna *j*. En otras palabras, se elimina la columna *j* si existe alguna columna *k* que tenga ceros, al menos, en todas las filas donde tiene la columna *j*. En este caso, tampoco se añade el ejemplo *j* a *S*.

4. Se repiten los pasos del uno al tres hasta que no se puedan hacer mayores progresos. Si no queda ninguna columna, el subconjunto *S* está completo y el algoritmo se acaba. Si queda alguna columna, se progresa al paso 5.

5. Hay que encontrar una fila *i* cuya inclusión en *S* conlleve la inclusión del menor número posible de otras filas. Para conseguir esto:

a) Para cada fila *i* de las que quedan, se asume que el ejemplo *i* va a añadirse a *S* y que, la fila *i* y cualquiera de las columnas que queden y tengan un 1 en esa fila *i*, serán borradas (aunque en realidad todavía no se borre ninguna de ellas). Suponiendo que se han eliminado, se encuentra el menor número de filas que habrá que tomar para tener al menos tantos 1 como columnas queden. A partir del mínimo calculado para cada fila *i*, se encuentra el mínimo absoluto para todas ellas.

b) Para cada fila *i* que en a) haya resultado con el mínimo número absoluto de filas adicionales necesitadas, se eliminan realmente la fila *i* y las columnas con un 1 en

la fila *i*, y se repite el proceso recursivamente empezando en el paso 1. Si se logra realmente el mínimo número de filas, se añade el ejemplo *i* a *S* y se para, ya que *S* está completo. En otro caso, se restaura la fila *i* y las columnas eliminadas, y se prueba la siguiente fila posible.

c) Si ninguna fila *i* tiene éxito en conseguir el mínimo número, se incrementa el mínimo absoluto y se intenta b) de nuevo hasta conseguir el éxito.

Puede apreciarse como en los pasos 2, 3 y 4 no se añade ningún ejemplo a *S*, sólo en los pasos 1 y 5 se añaden ejemplos al conjunto *S*. Este algoritmo también es sensible al ruido y tiende a salvaguardar la precisión sacrificando a cambio la reducción del conjunto de entrenamiento.

3.9.3. Regla del vecino más próximo reducida

Este algoritmo fue propuesto por Gates en 1972, se conoce como RNN (Reduced Nearest Neighbour Rule). Empieza con *S*=*T* y va eliminando cada ejemplo de *S*, siempre que esto no produzca un error en la clasificación de cualquier otro ejemplo de *T*, a partir de los ejemplos restantes de *S*.

Este algoritmo RNN es computacionalmente más costoso que el CNN de Hart, pero consigue un conjunto *S* menor que el de CNN y, por tanto, será menor el coste de almacenamiento y cálculo para cada nueva clasificación.

Además, este algoritmo, al no exigir que el ejemplo eliminado sea bien clasificado por el resto, será capaz de eliminar los ejemplos ruidosos y aquellos ejemplos situados en el interior de las regiones de decisión y, mantendrá principalmente los puntos fronterizos.

3.9.4. Regla del vecino más próximo editada

Fue introducida originalmente por Wilson en 1972, y posteriormente ha ido desarrollándose. La idea en la que se basa es eliminar ejemplos inmersos en agrupamientos de diferente clase o en las fronteras de las regiones de decisión.

El algoritmo ENN (Edited Nearest Neighbour Rule) empieza con S=T y se van eliminando aquellos ejemplos de S cuya clase no coincida con la clase mayoritaria de sus k-NN (donde k suele tomar el valor tres).

De esta forma se eliminan los ejemplos ruidosos y los ejemplos fronterizos, consiguiendo fronteras de decisión suavizadas. El inconveniente es que el ENN retiene casi todos los puntos internos, por lo que la reducción del conjunto de ejemplos a almacenar se ve seriamente perjudicada.

Por tanto, el método de editado es especialmente adecuado para eliminar los ejemplos ruidosos, pero no para reducir el conjunto a almacenar y, si se quiere conseguir esto, se aplicará después alguno de los otros métodos que consigan una importante reducción del conjunto a almacenar. Gráficamente el esquema sería:

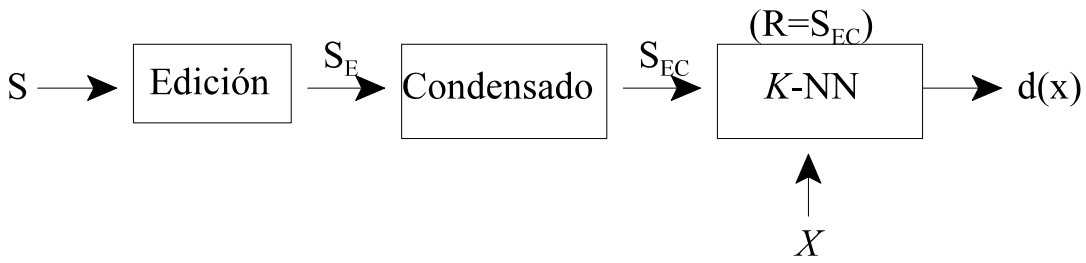


Figura 3.2 Esquema del proceso de editado y condensado del vecino más próximo.

3.9.5. Todos los *k*-NN

Este método es una extensión del ENN y fue desarrollando por Tomek en 1976. Recorre desde i=1 hasta i=k, señalando como malo cualquier ejemplo que no sea clasificado correctamente por sus i vecinos más próximos. Una vez realizadas las k iteraciones, se eliminan todos los ejemplos que han sido señalados como malos.

Este algoritmo consigue una mayor precisión que el ENN pero, al igual que él, no reduce significativamente el conjunto de entrenamiento, por tanto, como se acaba de decir, estos algoritmos son más útiles como filtros para los ejemplos ruidosos y para aumentar la precisión del clasificador, que para reducir el conjunto a almacenar.

En 1997, Kubat y Matwin intentan corregir esta situación modificando el algoritmo de Tomek para eliminar ejemplos internos además de los fronterizos. Primero aplican el CNN de Hart (o alguna variante) y después eliminan los ejemplos que participen en los vínculos de Tomek, es decir, pares de ejemplos de diferentes clases que se tengan como sus vecinos más próximos. Está preparado para manejar situaciones donde exista una clase mayoritaria y otra que sea mucho más rara de encontrar, en esta situación, sólo se eliminan los ejemplos de la clase mayoritaria, mientras que se retienen todos los ejemplos de la clase minoritaria.

3.9.6. Medida de similaridad entre variables

Este método fue introducido en 1995 por Lowe y se conoce como VSM (Variable Similarity Metric) que produce un nivel de confianza de sus clasificaciones.

Para conseguir eliminar los ejemplos ruidosos y reducir el conjunto de entrenamiento, un ejemplo se eliminará si todos sus k vecinos más próximos son de la misma clase, aunque ésta no coincida con la clase de dicho ejemplo (en este caso es probable que el ejemplo sea ruidoso). Así se consigue eliminar tanto los ejemplos ruidosos como los internos, y mantener los ejemplos fronterizos. Sin embargo, para

eliminar un ejemplo exige que sus vecinos tengan al menos un 60% de seguridad en su clasificación, este porcentaje puede variar en función de que se quiera realizar una reducción más agresiva o más conservadora. Si quiere ser más agresivo, se bajará el nivel de confianza exigido, mientras que, si se quiere ser más conservador, se exigirá una mayor seguridad en la clasificación.

Normalmente VSM se utiliza con valores grandes de *k* (por ejemplo *k*=10), lo que hace que la reducción sea bastante conservadora pero sí consigue mejorar la precisión de la clasificación. Además, el sistema VSM utiliza un voto ponderado por la distancia, por lo que resulta aconsejable utilizar un valor grande de *k*.

La siguiente tabla recoge, a modo de resumen, la forma de actuar de los métodos que se han visto, resaltando las principales diferencias entre ellos.

Métodos	Forma en que actúan
CNN	Cada miembro de *T* debe estar más cercano a un miembro de *S* de la misma clase que a cualquier miembro de *S* de una clase distinta.
SNN	Cada miembro de *T* debe estar más cercano a un miembro de *S* de la misma clase que a cualquier miembro de *T* de una clase distinta.
RNN	Empieza con *S*=*T* y va eliminando cada ejemplo de *S* siempre que esto no produzca un error en la clasificación de cualquier otro ejemplo de *T* a partir de los ejemplos restantes
ENN	Empieza con *S*=*T* y se van eliminando aquellos ejemplos de *S* cuya clase no coincida son la clase mayoritaria de sus *k*-NN (donde *k* suele tomar el valor tres).
Todos los *k*-NN.	Recorre desde *i*=1 hasta *i*=*k* señalando como malo cualquier ejemplo que no sea clasificado correctamente por sus *i* vecinos más próximos. Una vez realizada las *k* iteraciones se eliminan todos los ejemplos que han sido señalados como malos.
VSM	Son eliminados los ejemplos que todos sus *k* vecinos más próximos son de la misma clase, aunque esta no coincida con la clase de dicho ejemplo

Tabla 3.1 Métodos de reducción de la regla del vecino más próximo.

CAPÍTULO 4
MÉTODOS DE CLASIFICACIÓN INDIVIDUALES II: REDES NEURONALES ARTIFICIALES Y ARBOLES DE CLASIFICACIÓN

4.1. INTRODUCCIÓN

Este segundo capítulo referente a los métodos de clasificación individuales también se compone de dos grandes bloques. En el primero se estudian las redes neuronales artificiales y en el segundo, a partir del apartado 4.6, los árboles de clasificación.

En la primera parte de este capítulo se va a describir de forma sencilla en qué consisten los sistemas de redes neuronales artificiales. En concreto, se empezará analizando la unidad básica o uno de los modelos más sencillos de estos sistemas que es el perceptrón, estudiando también algunas de las reglas que se utilizan para

entrenarlo, la regla de entrenamiento del perceptrón y la regla delta. A continuación se estudiarán estructuras más complejas, como son las redes multicapa y el algoritmo que se suele utilizar para entrenarlas, el algoritmo de retropropagación de errores. Para acabar esta parte del capítulo se analizarán diversas cuestiones que hay que especificar a la hora de aplicar en la práctica las redes de neuronas artificiales.

Las Redes Neuronales Artificiales (RNA) son, en este contexto, un método no paramétrico de clasificación (entre otras muchas tareas), que permite obtener un grado elevado de precisión. Como se comentó en la sección 1.2, este trabajo se centra en el *aprendizaje supervisado* aunque también hay redes que utilizan aprendizaje de tipo *no supervisado*. En este sentido, se puede destacar los mapas auto-organizados de Kohonen que, aunque en principio puedan considerarse como un método de *aprendizaje no supervisado,* también puede utilizarse para la clasificación.

Las redes neuronales pretenden imitar el funcionamiento del cerebro humano en lo referente a su estructura neuronal que, de manera simplificada, constituye una red de neuronas interconectadas, lo que posibilita el procesamiento en paralelo. Así, en el caso de las redes neuronales artificiales, los nodos o unidades equivalentes a las neuronas reciben como entradas la suma ponderada de las salidas de otras unidades, que es como se cree que actúan las neuronas humanas.

Este sistema de aprendizaje tiene otras aplicaciones además de la clasificación, ya que su desarrollo se centra en el área de la Inteligencia Artificial y el Aprendizaje Automático. Es, probablemente, el método individual más complejo de los expuestos en este trabajo y, para el usuario final, resulta una especie de caja negra, ya que a partir de los datos de entrada obtendrá una clasificación, pero no una justificación de por qué se asigna esa clasificación, lo que lo hace más difícil de entender para el usuario.

Los ejemplos o casos están representados por pares atributo-valor, que servirán para describir el problema en cuestión. Estos atributos de entrada pueden estar correlacionados o no y pueden tomar cualquier valor real. Además de esos atributos,

cada ejemplo debe llevar una etiqueta con el valor de la función objetivo (en este caso la clase) que le corresponda. Esta función objetivo puede ser discreta, continua o, incluso, un vector de atributos continuos o discretos.

Una de las ventajas que presentan estos modelos es su comportamiento ante conjuntos de entrenamiento que puedan contener errores, ya que los sistemas de redes neuronales artificiales son bastante robustos a la presencia de ruido en estos conjuntos. Por otra parte, el tiempo empleado para entrenar la red suele ser elevado, sobre todo si se compara con el tiempo que necesitan otros métodos como, por ejemplo, los árboles de clasificación. Este tiempo dependerá de cuestiones como el número de elementos que haya que determinar con el aprendizaje de la red, el número de ejemplos de entrenamiento considerados y la elección de varios parámetros del algoritmo de aprendizaje. En cambio, el tiempo necesario para aplicar la red ya entrenada a un nuevo ejemplo es pequeño. Es decir, aunque el proceso de aprendizaje es lento, una vez entrenada la red, la evaluación de un nuevo caso es relativamente rápida.

Los árboles de clasificación son uno de los métodos no paramétricos de clasificación más utilizados en la actualidad, debido a su sencillez conceptual y a los buenos resultados que obtienen. La construcción de árboles se puede ver como un tipo de selección de variables en el que, cuestiones como la interacción entre variables o transformaciones monótonas de éstas, se manejan de forma automática. Los métodos basados en árboles pueden utilizarse también para tareas de regresión, donde cada nodo hoja es un valor predicho o una sencilla función, sin embargo se utilizan mayoritariamente como método de clasificación. La principal diferencia entre ambos es el tipo de variable dependiente que utilizan, el árbol de clasificación utiliza una variable cualitativa, mientras que para regresión se utiliza una variable cuantitativa.

En el apartado 4.6 se exponen algunas cuestiones de carácter general sobre los árboles. A continuación, se citan los pasos a seguir en la construcción del árbol para pasar a verlos , con mayor detenimiento, posteriormente. En primer lugar, se estudia la regla de corte o división, los criterios a seguir para realizar el corte pudiendo utilizar la

entropía o el indice de Gini para medir la impureza, la bondad de una partición, determinada por la reducción de impureza que consiga, y la impureza de un árbol, que depende de la impureza en sus nodos hoja. Además, hay que decidir el momento en que se detiene el desarrollo del árbol, por tanto, hay que establecer un criterio de parada.

En algunos casos el árbol puede crecer demasiado y adaptarse demasiado a las características particulares del conjunto de entrenamiento perdiendo capacidad de generalización, se dice entonces que el árbol está sobreajustado. Una solución al sobreajuste del árbol consiste en eliminar posteriormente alguno de sus nodos o ramas, es lo que se conoce como podado del árbol. Debido a la importancia de este proceso, existen diversas técnicas de podado, pudiendo hacerse la poda por mínimo coste-complejidad, o bien la poda de reglas. Por último, para acabar el capítulo se analiza el tratamiento que se hace de los valores omitidos, es decir, cuando en alguna observación no se dispone del valor en una o varias variables descriptivas.

4.2. MOTIVACIÓN BIOLÓGICA DE LAS REDES NEURONALES ARTIFICIALES

Como se acaba de decir, las redes neuronales artificiales están inspiradas en los sistemas de aprendizaje biológicos, que están compuestos por redes muy complejas de neuronas interconectadas. Así, las redes de neuronas artificiales también se componen de un conjunto interconectado más o menos denso de unidades sencillas, donde cada unidad toma un número de valores reales como entrada (generalmente las salidas de otras unidades) y produce un único valor real de salida (que puede ser utilizado como entrada de otras unidades).

Los neurobiólogos consideran que la habilidad para procesar información de los sistemas de neuronas biológicos se debe en buena medida a que éstos actúan mediante procesamiento en paralelo, utilizando representaciones que son distribuidas sobre muchas neuronas. El objetivo de los sistemas de redes neuronales artificiales es aprovechar esta clase de computación paralela basadas en representaciones distribuidas. Aunque los sistemas basados en redes neuronales artificiales estuvieron motivados por

los sistemas neuronales biológicos, hay muchas complejidades en éstos que las redes neuronales artificiales no han sido capaces de reproducir. Así mismo, hay características de las redes neuronales artificiales que no coinciden con los sistemas biológicos.

Históricamente dos grupos de investigadores han trabajado con redes neuronales artificiales. Uno de ellos con el objetivo de utilizar este tipo de redes para estudiar y modelizar los procesos de aprendizaje biológicos. Un segundo grupo motivado por el deseo de conseguir algoritmos de aprendizaje automático efectivos, sin importarles si estos algoritmos son o no iguales a los procesos biológicos. En este trabajo interesa más este segundo grupo.

4.3. EL PERCEPTRÓN SIMPLE

Una de las unidades básicas en la que se basan algunos de los sistemas de redes neuronales artificiales es lo que se conoce como perceptrón. Un perceptrón calcula una combinación lineal de los valores del vector de entradas y, si esa combinación lineal supera un determinado umbral, entonces asigna el valor 1, mientras que, si no lo supera, asignará el valor -1. De forma más precisa, si se considera el vector de entradas $\vec{x} = (x_1, x_2, ..., x_n)$, la salida $\vec{y} = f(x_1, x_2, ..., x_n)$ que calcula el perceptrón es

$$y = \begin{cases} 1 & si \sum_{i=1}^{n} x_i w_i > \theta \\ -1 & si \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \quad \textbf{(4.1)}$$

donde cada w_i es una constante real que se suele llamar peso sináptico o simplemente peso y que determina la aportación del valor de entrada x_i a la salida del perceptrón. θ es el umbral que debe sobrepasar la combinación lineal ponderada de las entradas, $w_1x_1 + w_2x_2 + ... + w_nx_n$, para que la salida del perceptrón sea 1.

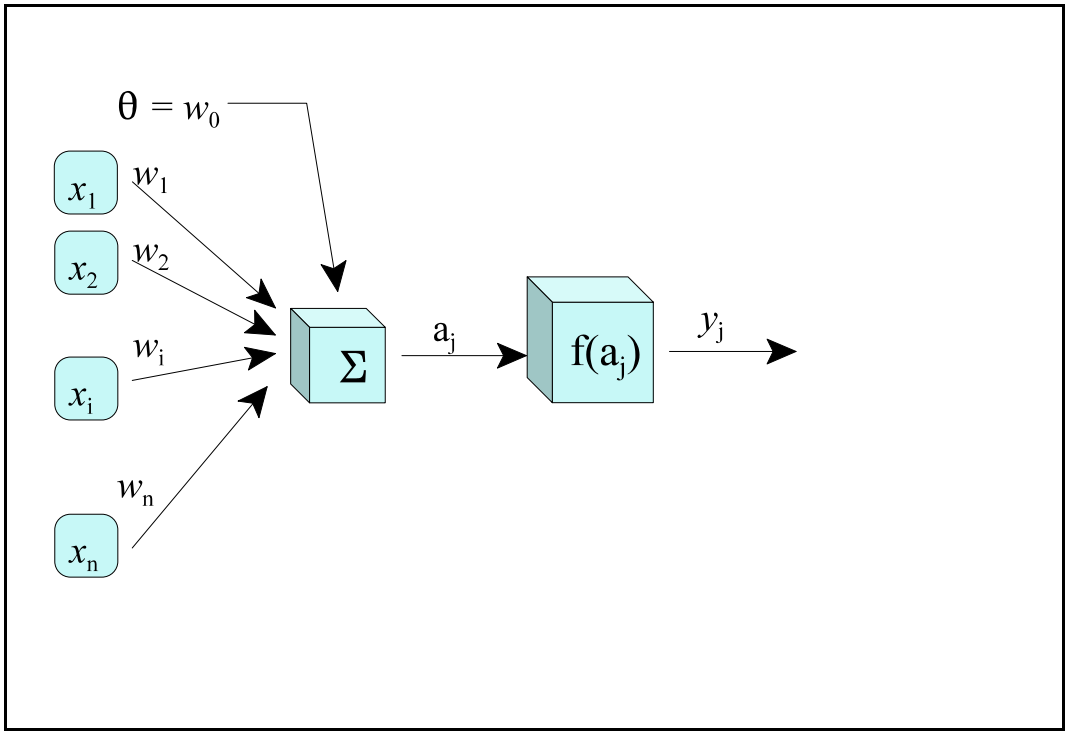


Figura 4.1 Esquema del perceptrón simple

En la figura anterior se puede observar la estructura del perceptrón simple, donde a_j se denomina *entrada neta* y se calcula como $a_j = \sum_{j=1}^{n} w_j x_j$ y f es lo que se conoce como función de transferencia o activación, que en este caso es de tipo signo, aunque existen otras posibilidades como se verá más adelante. Normalmente para simplificar la notación se añade un 1 al vector de entradas ($x_0 = 1$) y, considerando w_0=**2**, se puede escribir la desigualdad anterior como $\sum_{i=0}^{n} x_i w_i > 0$ o, en forma vectorial, como $\vec{w}\cdot\vec{x}>0$.

Algunas veces la función del perceptrón se representa por $y = sign(\vec{w}\cdot\vec{x})$ donde

$$y = sign(\vec{w}\cdot\vec{x}) = \begin{cases} 1 & si \sum_{i=0}^{n} x_i w_i > 0 \\ -1 & si \sum_{i=0}^{n} x_i w_i < 0 \end{cases} \qquad \textbf{(4.2)}$$

El entrenamiento del perceptrón consiste en la elección de los valores adecuados para los pesos w_0 , w_1 ..., w_n. Por tanto, el espacio H de posibles hipótesis consideradas en el aprendizaje del perceptrón es el conjunto de todos los posibles vectores de pesos con valores reales $H = \{\vec{w} / \vec{w} \in R^{n+1}\}$

El perceptrón puede verse como un hiperplano que constituye una superficie de decisión en el espacio n-dimensional de ejemplos[5]. El perceptrón asigna un 1 para los ejemplos que caen a un lado del hiperplano y un -1 para los ejemplos que caen al otro lado. La ecuación para este plano de decisión es $\vec{w} \cdot \vec{x} = 0$. Algunos conjuntos de ejemplos de distinta clase no podrán ser separados totalmente por ningún hiperplano. Aquéllos que pueden ser separados, se llaman conjuntos de ejemplos linealmente separables. Por tanto, el principal problema del perceptrón simple es su limitación para resolver problemas que no son linealmente separables.

4.3.1. La regla de entrenamiento del perceptrón

El entrenamiento del perceptrón consiste en encontrar un vector de pesos $\vec{w}$ que consiga que el perceptrón obtenga la salida 1 ó -1 correcta para cada ejemplo del conjunto de entrenamiento. Este problema de aprendizaje puede resolverse mediante varios algoritmos. Este trabajo se centrará en dos, la regla del perceptrón y la regla delta o de gradiente descendente. Estos algoritmos convergen a hipótesis aceptables en ambos casos, aunque ligeramente distintas entre sí, debido a que las condiciones bajo las que trabajan ambos algoritmos son algo diferentes. Su importancia no radica únicamente en la resolución del problema de entrenar el perceptrón, sino en que constituyen la base para el aprendizaje de modelos de redes más complejos.

Una manera de obtener un vector de pesos aceptable es empezar con pesos aleatorios y, posteriormente, aplicar el perceptrón a cada ejemplo de entrenamiento iterativamente, modificando los pesos del perceptrón siempre que se equivoque en la clasificación de

[5] Cada ejemplo supondría un punto en ese espacio n-dimensional.

un ejemplo. Este proceso se repite una y otra vez hasta que el perceptrón clasifique correctamente todos los ejemplos de entrenamiento. Los pesos se modifican en cada paso de acuerdo a la regla de entrenamiento del perceptrón, de tal forma que cada nuevo peso se calcula corrigiendo el peso antiguo. En concreto, si se representa el peso actual como w_i(t), que sería el peso asociado al atributo x_i en el periodo *t*. El nuevo peso será $w_i(t+1)$ que será el peso correspondiente a ese mismo atributo en el periodo $t+1$. El nuevo peso, $w_i(t+1)$, se calcula mediante un factor de actualización, $\Delta w_i(t)$, aplicado sobre el peso actual, $w_i(t)$. La actualización también incluye al umbral $\theta(t)$ que se ha igualado a $w_0(t)$ para considerarlo como un peso más.

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \qquad \textbf{(4.3)}$$

Como se ha dicho, la actualización de los pesos sólo se lleva a cabo cuando la clasificación que proporciona el perceptrón es incorrecta, por lo que se dice que el entrenamiento del perceptrón es un procedimiento de corrección del error. Además, esta actualización se realiza tras la presentación de cada ejemplo individualmente, y no conjuntamente una vez clasificados todos los ejemplos del conjunto de entrenamiento. Ahora bien, el factor de actualización deberá tener en cuenta el error cometido en la clasificación por el perceptrón, en concreto este factor toma el valor

$$\Delta w_i(t) = (y^d - y)x_i \qquad \textbf{(4.4)}$$

donde y^d es la clasificación deseada e *y* es la salida del perceptrón. Esta ecuación es coherente con la idea de actualizar solamente cuando se produce un error en la clasificación, ya que si $y^d = y$ entonces $\Delta w_i(t)=0$ y $w_i(t+1) = w_i(t)$.

Cuando el perceptrón asigna un -1 y el verdadero valor es 1, para conseguir que la salida del perceptrón sea 1 en lugar de -1, habrá que modificar los pesos para aumentar el valor de la combinación lineal de pesos y valores. Por ejemplo, si $x_i > 0$, entonces aumentando w_i se acercará el perceptrón a la clasificación correcta del ejemplo. Efectivamente, la regla de entrenamiento aumentará w_i en ese caso ya que $(y^d - y)$ y x_i son

ambos positivos. Por último si y^d = -1 e y=1, entonces los pesos asociados con x_i positivos serán reducidos en lugar de incrementados.

Cuando las clases sean linealmente separables, el proceso de entrenamiento del perceptrón convergerá a la respuesta correcta. Es decir, si existe un discriminante lineal capaz de separar las clases sin cometer ningún error, el proceso de entrenamiento encontrará el hiperplano de separación[6].

Aunque el teorema de convergencia del perceptrón asegura que para datos linealmente separables el perceptrón convergerá, el tiempo necesario para el aprendizaje no se sabe. Puede que el proceso tarde mucho tiempo en converger, ya que el perceptrón no necesariamente se estará acercando a la solución después de cada ejemplo. Sin embargo, existen algunas modificaciones sobre los datos y sobre el proceso de entrenamiento que pueden acelerar el aprendizaje y la convergencia, como por ejemplo:

- Normalizar los datos, muchas veces las distintas variables vendrán medidas en diferentes escalas pudiendo llegar a haber grandes diferencias entre los valores que toma una variable y los de otra. En ese caso se puede mejorar el comportamiento simplemente normalizando los datos antes del entrenamiento, de tal forma que todas las variables varíen entre 0 y 1.

- Introducir una constante positiva, llamada tasa de aprendizaje y que se representará por η, en el cálculo del factor de actualización de los pesos del perceptrón. Normalmente la tasa de aprendizaje también toma valores entre 0 y 1. La función de la tasa de aprendizaje es moderar el grado en que los pesos son modificados en cada paso.

Si los ejemplos son linealmente separables, la tasa de aprendizaje puede ser cualquier número real positivo y la convergencia estará asegurada. En cada iteración, una vez

[6] En los casos particulares de sólo dos o tres dimensiones, en lugar de un hiperplano las fronteras de decisión las marcarán una linea o un plano, respectivamente.

presentados todos los ejemplos del conjunto de entrenamiento, se espera que el perceptrón se acerque cada vez más a la solución y, por tanto, tenga que realizar ajustes cada vez más pequeños en los pesos. Por ello, es habitual hacer la tasa de aprendizaje inversamente proporcional al tiempo ya empleado en el aprendizaje, o sea $\eta = 1/t$. Es decir, la tasa de aprendizaje disminuirá conforme aumente el número de iteraciones empleadas en el ajuste de los pesos. En realidad, a priori no se sabe el valor de la tasa de aprendizaje que acelerará al máximo la convergencia para un conjunto de datos determinado.

Si se modifica la ecuación (4.4) para introducir la tasa de aprendizaje, quedará

$$\Delta w_i(t) = \eta(y^d - y)x_i \qquad \textbf{(4.5)}$$

En este caso el factor de actualización está compuesto por el error cometido, por el valor de entrada y por la tasa de aprendizaje.

Por tanto, la regla de entrenamiento del perceptrón asegura la convergencia a un vector de pesos que clasifique correctamente todos los ejemplos, si los ejemplos de entrenamiento son linealmente separables y utilizando un ratio de aprendizaje suficientemente bajo (véase Minsky y Papert, 1969). Si los ejemplos no son linealmente separables, el uso de una tasa de aprendizaje que disminuya con el tiempo puede proporcionar resultados aceptables, aún en los casos en los que la convergencia total sea imposible. Debido a las dificultades de la regla de entrenamiento del perceptrón ante clases no linealmente separables, lo que es bastante frecuente, existe otra regla de entrenamiento, llamada regla delta, que se estudia a continuación.

4.3.2. La regla delta o del gradiente descendente

Aunque la regla del perceptrón funciona bien cuando los ejemplos de entrenamiento son linealmente separables, como se ha dicho puede no encontrar un vector de pesos adecuado cuando los ejemplos no son linealmente separables. Para resolver este

problema se puede utilizar la regla de entrenamiento que se conoce como regla delta. Si los ejemplos de entrenamiento no son linealmente separables, la regla delta converge hacia una aproximación más ajustada a la función objetivo.

La idea fundamental de la regla delta es utilizar el gradiente descendente para buscar en el espacio de hipótesis de los posibles vectores de pesos y encontrar así, los pesos que mejor se ajustan a los ejemplos de entrenamiento. Esta regla es importante porque el gradiente descendente sirve de base para el algoritmo de retropropagación, que puede entrenar redes de varias capas de unidades interconectadas.

Para entender mejor la regla delta considérese el problema de entrenar un perceptrón sin umbral, es decir, una unidad lineal para la que la salida *y* viene dada por

$$y(\vec{x}) = \vec{w}\,\vec{x} \tag{4.6}$$

que puede verse como un primer paso del perceptrón, sin el umbral.

La regla delta también es un procedimiento de corrección del error. Para medir el error que se comete utiliza la siguiente medida:

$$E(\vec{w}) = \frac{1}{2}\sum_{p\in P}(y_p^{\,d} - y_p)^2 \tag{4.7}$$

donde P es el conjunto de ejemplos o patrones de entrenamiento, $y_p^{\,d}$ es la salida deseada para el ejemplo p e y_p es la salida de la unidad lineal para ese ejemplo. Es decir, el error de entrenamiento se calcula como la mitad del cuadrado de la diferencia entre el valor deseado y el valor obtenido para la salida de cada ejemplo sumado sobre todos los ejemplos de entrenamiento. La notación $E(\vec{w})$ hace hincapié en que el error cometido depende del vector de pesos y, aunque también influya el conjunto de entrenamiento en cada caso, éste permanecerá constante durante el entrenamiento. Por tanto, lo que interesa es modificar esos pesos de forma que se minimice el error.

Para entender mejor el algoritmo del gradiente descendente, puede resultar muy útil visualizar el espacio de hipótesis de los posibles vectores de pesos y los valores de los errores cometidos en cada caso. Para ello, Mitchell (1997) utiliza el caso simplificado de un vector de pesos bidimensional, donde el plano determinado por los pares $\{w_0, w_1\}$ representa el espacio completo de hipótesis. En el eje vertical se recoge el error cometido para cada par de pesos $\{w_0, w_1\}$ en un conjunto de entrenamiento determinado. La figura 4.2 muestra la superficie de error, que expresa la idoneidad de cada vector de pesos en el espacio de hipótesis (interesa minimizar el error). Debido a la forma en que está definido el error, esa superficie será parabólica con un mínimo global. La parábola concreta dependerá del conjunto de entrenamiento del problema en cuestión.

La búsqueda del gradiente descendente determina un vector de pesos que minimice el error, empezando con un vector de pesos inicial arbitrario y modificando repetidamente en pequeños pasos. En cada paso, el vector de pesos se actualiza en la dirección que produzca el mayor descenso a lo largo de la superficie del error, como indica la flecha de la figura 4.2. El proceso continúa hasta que se alcanza el error mínimo global.

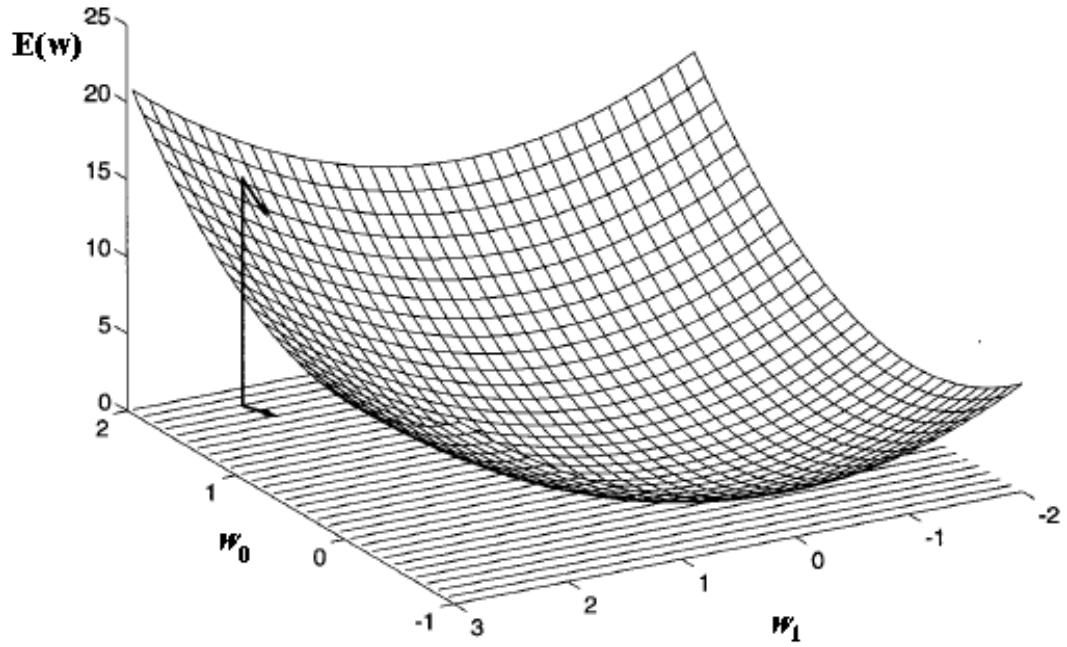


Figura 4.2. Superficie del error en función de los pesos.

Aunque el ejemplo expuesto es para el caso simplificado en el que sólo hay que ajustar dos pesos, la misma idea es aplicable al caso general de un vector de pesos n-dimensional. Hay que determinar cómo calcular la dirección que produzca un mayor descenso a lo largo de la superficie del error. Para ello se calculará la derivada del error con respecto a cada componente del vector de pesos, lo que dará lugar a un vector compuesto por las derivadas parciales, que se conoce como el gradiente del error con respecto al vector de pesos, y que se representa por $\Delta E(\vec{w})$ y se calcula como

$$\Delta E(\vec{w}) = \left\{ \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right\} \qquad \textbf{(4.8)}$$

Si se interpreta como un vector en el espacio de pesos, el gradiente especifica la dirección que produce el mayor aumento en el error. Por tanto, el opuesto a este vector dará la dirección del mayor descenso. Por ejemplo, la flecha en la figura 4.2 muestra el gradiente negativo $-\Delta E(\vec{w})$ para un determinado punto en el plano $\{w_0, w_1\}$.

Como el gradiente descendente indica la dirección del mayor descenso de E, la regla de entrenamiento para el gradiente descendente es también $\vec{w}(t+1)=\vec{w}(t)+\Delta\vec{w}(t)$, donde $\Delta\vec{w} = -\eta\, \Delta E(\vec{w})$.

De nuevo η es el parámetro de tasa de aprendizaje, una constante positiva que determina el tamaño del paso en la búsqueda del gradiente descendente. El signo negativo lleva a mover los pesos en la dirección en la que disminuye el error. Esta regla de entrenamiento puede escribirse también en forma de las componentes $w_i(t+1) = w_i(t) + \eta\, w_i(t)$ donde

$$\Delta w_i(t) = -\eta \frac{\partial E}{\partial w_i} \qquad \textbf{(4.9)}$$

que pone de manifiesto que el mayor descenso se consigue modificando, cada componente w_i del vector de pesos, de forma proporcional a la derivada parcial del error respecto de esa componente.

Para construir un algoritmo práctico que permita actualizar iterativamente los pesos de acuerdo a la ecuación (4.9), se necesita un modo eficiente de calcular el gradiente en cada paso. El vector de las derivadas parciales que forma el gradiente puede obtenerse derivando el error a partir de la ecuación (4.7)

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\frac{1}{2}\sum_{p\in P}(y_p^d - y_p)^2\right) = \frac{1}{2}\sum_{p\in P}\frac{\partial}{\partial w_i}(y_p^d - y_p)^2$$

$$= \frac{1}{2}\sum_{p\in P} 2(y_p^d - y_p)\frac{\partial}{\partial w_i}(y_p^d - y_p) = \sum_{p\in P}(y_p^d - y_p)\frac{\partial}{\partial w_i}(y_p^d - \vec{w}\vec{x}_p) \qquad \textbf{(4.10)}$$

$$\frac{\partial E}{\partial w_i} = \sum_{p\in P}(y_p^d - y_p)(-x_{ip})$$

donde x_{ip} representa el valor de la componente x_i para el ejemplo p. Si se sustituye este resultado en la ecuación 4.9 se obtiene la regla de actualización de los pesos para el gradiente descendente

$$\Delta w_i(t) = \eta\sum_{p\in P}(y_p^d - y_p)x_{ip} \qquad \textbf{(4.11)}$$

Si el valor de la tasa de aprendizaje es demasiado grande, este algoritmo corre el riesgo de sobrepasar el mínimo en la superficie del error, en lugar de caer en él. Por esta razón, como ya se ha dicho, es común reducir el valor de η conforme aumenta el número de pasos.

Las principales dificultades en la aplicación del gradiente descendente son:

1. La convergencia a un mínimo global puede a veces ser muy lenta, en algunos casos incluso necesita miles de iteraciones.

2. Si hay varios mínimos locales en la superficie del error, no se puede garantizar que el procedimiento encuentre el mínimo global.

Para evitar esto se puede utilizar una variación del gradiente descendente estándar que se conoce como gradiente descendente incremental, estocástico u *on-line*. Este método, en lugar de actualizar los pesos después de sumar los errores sobre todos los ejemplos de entrenamiento, lo que hace es actualizar los pesos incrementalmente calculando el error para cada ejemplo.

La regla de entrenamiento modificada es similar a la de la ecuación (4.11), salvo porque al actualizar después de cada ejemplo la actualización de los pesos se calcula en cada paso como

$$\Delta w_i(t) = \eta (y^d - y) x_i \tag{4.12}$$

En realidad también se ha modificado la función de error, que ahora se representa y calcula como

$$E_p(\vec{w}) = \frac{1}{2}(y_p^d - y_p)^2 \tag{4.13}$$

La secuencia de estas actualizaciones de los pesos, cuando se repiten sobre todos los ejemplos del conjunto de entrenamiento, aproxima razonablemente el descenso del gradiente con respecto a la función original $E(\vec{w})$. Haciendo el valor de η suficientemente pequeño, el gradiente descendente *on-line* puede aproximar al verdadero valor del gradiente descendente.

Aunque tanto el método *on-line* como el estándar son utilizados, los motivos para preferir el algoritmo incremental según Ripley (1996, pág 154), son principalmente tres. El primero es la motivación biológica de aprender de cada experiencia. Otro es que puede converger más rápidamente que la versión estándar, suponiendo que el conjunto de entrenamiento contiene ejemplos duplicados o muy similares. Por último, si existen ejemplos ruidosos o mal clasificados el algoritmo *on-line* que va utilizando los ejemplos de forma aleatoria es más probable que evite un mínimo local en la búsqueda del error mínimo.

4.4. REDES MULTICAPA

El perceptrón simple expresa superficies de decisión lineales. Sin embargo, muchas veces interesa poder representar funciones de todo tipo, distintas a la lineal. Para ello se va a utilizar las *redes multicapa*, que además de las *capas de entrada* y *salida*, presentan una o más capas intermedias que se conocen como *capas ocultas* y que permiten resolver problemas no lineales. La capa de entrada presenta tantas unidades como variables haya para describir a cada individuo. La capa de salida está compuesta por tantos elementos como el número de posibles clases. Por último, el número de capas ocultas y la estructura de las mismas (número de nodos en cada una) habrá que determinarlo antes del entrenamiento, aunque en la práctica una o dos capas ocultas suelen ser suficientes.

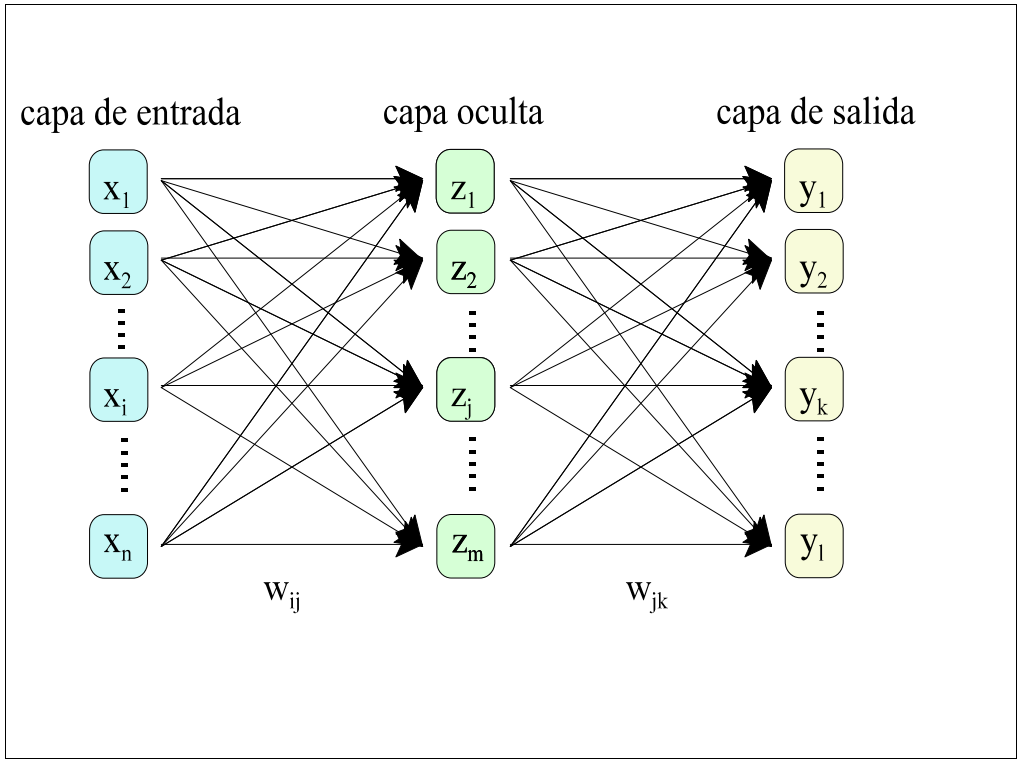


Figura 4.3 Red Multicapa.

Las redes multicapa entrenadas por el algoritmo de retropropagación[7] son capaces de expresar una gran variedad de superficies de decisión no lineales, lo que les permite ajustarse mucho mejor a determinadas aplicaciones reales.

Antes de continuar conviene aclarar la notación que se va a utilizar:

• x_i, i= 0, 1, .., n; z_j, j= 0, 1, .., m; y_k, k= 1, .., l son los valores de los nodos de la capa de entrada, oculta y de salida, respectivamente.

• w_{ij} es el peso del nodo i de entrada en el nodo j de la capa oculta y w_{jk} el del nodo j en la unidad de salida k.

• u_j y a_k son las entradas netas de la capa oculta y de salida, respectivamente. Y se calculan como $u_j = \sum_{i=0}^{n} w_{ij} x_i$ y $a_k = \sum_{j=0}^{m} w_{jk} z_j$

• y_k^d es el valor deseado para el nodo de salida k.

Como unidad básica para construir las redes multicapa no se van a elegir las unidades lineales estudiadas en la sección anterior (4.3), porque si se utilizan múltiples capas de unidades lineales en cascada también producirán funciones lineales, y se buscan redes que puedan representar funciones distintas a la lineal. Tampoco sirve la función del perceptrón, porque al ser discontinua no es diferenciable y, por tanto, no es adecuada para la búsqueda del gradiente descendente. Se necesita, en consecuencia, una expresión cuya salida sea una función no lineal de sus entradas, pero sí diferenciable de éstas. Una opción muy utilizada es la función sigmoidea, que es una función suave, diferenciable y limitada entre 0 y 1. Como puede verse en la figura 4.4, la función sigmoidea, para

[7] El algoritmo de retropropagación es conocido en inglés como backpropagation y en muchas ocasiones se utilizan las siglas BP para representarlo.

cualquier valor que tomen sus entradas netas, que se representarán por $a = \vec{w}\vec{x} = \sum_{i=0}^{n} w_i x_i$, devuelve un valor entre 0 y 1. En concreto, la salida será

$$y = \frac{1}{1+e^{-a}} \qquad \textbf{(4.14)}$$

Además, esta función es fácilmente derivable en relación a las entradas *a*, ya que es $\frac{\partial y}{\partial a} = y(1-y)$, lo que será utilizado por la regla de aprendizaje del gradiente descendente.

Esta función sigmoidea se puede modificar fácilmente introduciendo una constante *k*, quedando e^{-ka} en lugar de e^{-a}, donde *k* es una constante positiva que determina la suavidad de la función y que no afecta a la sencillez en el cálculo de la derivada.

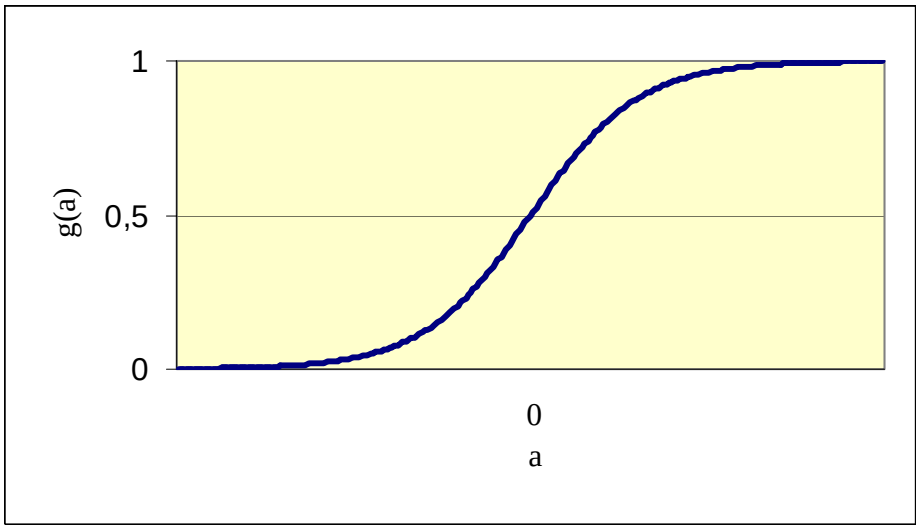


Figura 4.4 Función sigmoidea

4.4.1. El algoritmo de retropropagación

Este algoritmo ajusta los pesos para una red con un número fijo de unidades o interconexiones. Se utiliza el gradiente descendente para intentar minimizar el error cuadrático entre los valores de salida de la red y los valores reales de esas salidas.

En primer lugar, habrá que redefinir el error ya que, en general, se hablará de redes con varias unidades de salida y, por tanto, hay que sumar los errores sobre todas las unidades de salida de la red.

$$E(\vec{w}) = \frac{1}{2}\sum_{p\in P}\sum_{k=1}^{l}(y_{pk}^{d} - y_{pk})^2 \qquad \textbf{(4.15)}$$

donde l es el número de unidades de salida en la red e y_{pk}^{d} e y_{pk} son, respectivamente, los valores deseados y obtenidos asociados con la unidad de salida k y el ejemplo de entrenamiento p.

El problema de aprendizaje planteado por el algoritmo de retropropagación es buscar en un espacio de hipótesis muy grande, definido por todas las combinaciones de los posibles valores de los pesos de todas las unidades de la red. Puede plantearse como una superficie de error en función del vector de pesos, de tal forma que se puede utilizar el gradiente descendente para intentar encontrar un vector de pesos que minimice el error.

En el caso de las redes multicapa la superficie del error puede tener varios mínimos locales, lo que hace que el gradiente descendente garantice la convergencia hacia algún mínimo local, pero no necesariamente hacia el error mínimo global. A pesar de este inconveniente, en la práctica, el algoritmo de retropropagación ha mostrado resultados muy buenos en muchas aplicaciones del mundo real.

Para simplificar se va a utilizar una red con una sola capa oculta y con todas las conexiones hacia delante, las unidades de cada capa están conectadas a todas las

unidades de la capa siguiente. Además se va a manejar la versión *on-line* o incremental del gradiente descendente.

Antes de aplicar el algoritmo de retropropagación hay que construir la red con el número deseado de unidades ocultas y de salida, así como elegir el número de capas ocultas, e inicializar todos los pesos de la red en valores aleatorios pequeños. Durante el algoritmo, la estructura de la red permanece fija y los pesos se irán actualizando tras la presentación de cada ejemplo y según el error cometido por la red en ese ejemplo y el gradiente con respecto al error en ese ejemplo. Este paso del gradiente descendente es repetido hasta que la red se comporte aceptablemente bien. A veces son necesarias miles de repeticiones, utilizando los mismos ejemplos de entrenamiento varias veces.

El nombre de retropropagación se debe a que el error cometido se lleva hacia atrás capa a capa (en este ejemplo se trabaja con una sola, pero se puede generalizar a varias capas ocultas). Por tanto, a la hora de actualizar se empezará por la capa de unidades de salida y, después, se continuará con las capas ocultas. El tratamiento es diferenciado, ya que los ejemplos de entrenamiento sólo proporcionan valores objetivo para las unidades de salida, y no hay valores objetivo disponibles para calcular directamente el error de las unidades ocultas. Lo que se hace es sumar los errores cometidos por todas las unidades de salida influidas por cada unidad oculta j, ponderándolos por los pesos $w_{j\,k}$, el peso de la unidad *j* en la unidad *k*. Este peso representa el grado de responsabilidad de la unidad oculta *j* en el error cometido por la unidad de salida *k*.

4.4.2. Derivación de la regla de retropropagación

Si se utiliza el gradiente descendente incremental, se actualizará después de cada ejemplo de entrenamiento *p*, descendiendo por el gradiente del error E_p con respecto a ese ejemplo. Es decir, para cada ejemplo de entrenamiento *p* todos los pesos w_{ij} son actualizados añadiéndoles el factor

$$\Delta w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}} \qquad \textbf{(4.16)}$$

donde E_p es el error en el ejemplo p sumado sobre todas las unidades de salida en la red y se calcula como

$$E_p(\vec{w}) = \frac{1}{2}\sum_{k=1}^{l}(y_k^d - y_k)^2 \quad \textbf{(4.17)}$$

Como se ha dicho, se comienza con la actualización de los pesos de los nodos de salida. Se van a derivar las expresiones $\frac{\partial E_d}{\partial w_{ij}}$ y $\frac{\partial E_d}{\partial w_{jk}}$ para poder aplicar la regla del gradiente descendente incremental de la ecuación (4.16). Para empezar, se debe tener en cuenta que el peso w_{jk} sólo puede influir en el error por medio de a_k. Por tanto, se puede utilizar la regla de la cadena para escribir

$$\frac{\partial E_p}{\partial w_{jk}} = \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial w_{jk}} = \frac{\partial E_p}{\partial a_k} z_j \quad \textbf{(4.18)}$$

Ahora bien, a_k influirá en el error sólo a través de y_k. Por tanto, se puede utilizar de nuevo la regla de la cadena para escribir

$$\frac{\partial E_p}{\partial a_k} = \frac{\partial E_p}{\partial y_k}\frac{\partial y_k}{\partial a_k} \quad \textbf{(4.19)}$$

Si se considera en primer lugar el primer término de la ecuación (4.19)

$$\frac{\partial E_p}{\partial y_k} = \frac{\partial}{\partial y_k}\left(\frac{1}{2}\sum_{k=1}^{l}(y_k^d - y_k)^2\right)$$

las derivadas $\frac{\partial}{\partial y_k}(y_k^d - y_k)^2$ serán cero para todas las unidades de salida, salvo para la unidad k. Por tanto, se elimina el sumatorio y queda

$$\frac{\partial E_p}{\partial y_k} = \frac{1}{2}\frac{\partial}{\partial y_k}(y_k^d - y_k)^2 = \frac{1}{2}2(y_k^d - y_k)\frac{\partial(y_k^d - y_k)}{\partial y_k} = -(y_k^d - y_k) \quad \textbf{(4.20)}$$

Por otra parte, el segundo término de la ecuación (4.19) es la derivada de la función sigmoidea respecto a su entrada a_k, que ya se ha visto que es

$$\frac{\partial y_k}{\partial a_k} = y_k (1 - y_k) \tag{4.21}$$

Si se sustituyen los resultados de las derivadas en (4.20) y (4.21) en la expresión (4.19), se tiene

$$\frac{\partial E_p}{\partial a_k} = -(y_k^d - y_k)\, y_k (1 - y_k) \tag{4.22}$$

y, recordando el resultado obtenido en (4.18), se puede sustituir finalmente en (4.16), obteniendo la regla del gradiente descendente incremental para las unidades de salida

$$\Delta w_{jk} = -\eta \frac{\partial E_p}{\partial w_{jk}} = \eta (y_k^d - y_k)\, y_k (1 - y_k)\, z_j = \eta\, \delta_k\, z_j \tag{4.23}$$

donde para simplificar se ha igualado $\delta_k = (y_k^d - y_k)\, y_k (1 - y_k)$

Para las unidades ocultas la actualización de los pesos se llevará a cabo aplicando también la regla de la cadena

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial u_j} \frac{\partial u_j}{\partial w_{ij}} = \frac{\partial E_p}{\partial u_j} x_i \tag{4.24}$$

La entrada neta u_j sólo puede influir en el error a través de las entradas netas de las unidades de la capa de salida, que se ven afectadas por la salida de la unidad oculta *j*. En principio, se va a suponer que afecta a todas las unidades de esta capa.

$$\begin{aligned} \frac{\partial E_p}{\partial u_j} &= \sum_{k=1}^{l} \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial u_j} = \sum_{k=1}^{l} -\delta_k \frac{\partial a_k}{\partial u_j} = \sum_{k=1}^{l} -\delta_k \frac{\partial a_k}{\partial z_j} \frac{\partial z_j}{\partial u_j} = \sum_{k=1}^{l} -\delta_k w_{jk} \frac{\partial z_j}{\partial u_j} \\ &= \sum_{k=1}^{l} -\delta_k w_{jk} z_j (1 - z_j) \end{aligned} \tag{4.25}$$

Sustituyendo en la ecuación (4.16) los resultados de (4.24) y (4.25), se obtiene

$$\Delta w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}} = \eta z_j(1-z_j)x_i \sum_{k=1}^{l} \delta_k w_{jk} = \eta \delta_j x_i \qquad \textbf{(4.26)}$$

que es la regla de actualización de los pesos de las unidades de las capas ocultas, donde para simplificar se ha hecho $\delta_j = z_j(1-z_j)\sum_{k=1}^{l} \delta_k w_{jk}$

Aunque se ha calculado para una red con una única capa oculta, la generalización a una red de *h* capas intermedias es sencilla. La actualización de los pesos de las diferentes capas se realiza hacia atrás, repitiendo el procedimiento y calculando de manera iterativa los correspondientes términos $*_h$.

4.4.3. Inclusión del impulso

Una de las modificaciones más frecuentes del algoritmo de retropropagación consiste en transformar la regla de actualización de los pesos, haciendo que esta actualización en la iteración *t* dependa en parte de la actualización llevada a cabo en la iteración inmediatamente anterior, *t*-1, como expresa la siguiente ecuación.

$$\Delta w_{ij}(t) = \eta \delta_j x_i + \alpha \Delta w_{ij}(t-1) \qquad \textbf{(4.27)}$$

donde $) w_{ij}(t)$ y $) w_{ij}(t\text{-}1)$ son las actualizaciones del peso w_{ij} durante la iteración *t* y *t*-1 respectivamente y " es una constante que varía entre 0 y 1. De los dos términos que aparecen en la derecha de esta ecuación, el segundo se llama término impulso. Según Mitchell (1997) el efecto de este término puede entenderse mejor considerando que la trayectoria de la búsqueda del gradiente descendente es como un balón que rueda descendiendo por la superficie del error. El término de impulso tiende a mantener el balón rodando en la misma dirección de una iteración a la siguiente. Esto puede

conseguir que el balón ruede a través de un mínimo local (sin quedarse en él) o a lo largo de una región plana en la superficie, donde el balón se detendría si no estuviese el impulso. Además, aumenta gradualmente el tamaño del paso de la búsqueda en regiones donde el gradiente no cambia, por lo que acelera la convergencia.

4.5. LA APLICACIÓN PRÁCTICA DEL ALGORITMO DE RETROPROPAGACIÓN

Al igual que en otros métodos de clasificación hay que realizar varias elecciones a la hora de aplicar este algoritmo. Pequeñas variaciones pueden determinar que el comportamiento sea bueno o malo, en términos de precisión. Aunque se va a considerar algunas de esas variaciones, en realidad no se puede afirmar qué es mejor de forma general e irrevocable, y la única forma de encontrar los mejores resultados será repetir el aprendizaje modificando las condiciones experimentales. Algunas de las opciones a determinar son las siguientes:

1. Estado inicial aleatorio. El valor inicial de los pesos se suele asignar aleatoriamente entre -0,5 y 0,5. Incluso con las mismas condiciones de aprendizaje, los pesos aleatorios iniciales pueden llevar a resultados distintos de una sesión de entrenamiento a otra. En general se repite el proceso varias veces para obtener el mejor resultado posible.

2. Actualizar por iteración o por caso. Como ya se ha comentado, la actualización de los pesos puede realizarse una vez presentados todos los ejemplos del conjunto de entrenamiento teniendo en cuenta la suma de los errores cometidos, o bien, después de la presentación de cada ejemplo. Aunque la actualización por iteración tenga un mayor fundamento teórico, la revisión *on-line* puede lograr mejores resultados y se utiliza más en la práctica.

3. Presentación secuencial o aleatoria. La iteración es la unidad fundamental del aprendizaje y, muchas veces, la longitud de éste se mide por el número de iteraciones. Ahora bien, cuando se actualiza *on-line,* durante cada iteración los casos pueden ser

presentados en el mismo orden, lo que se conoce como secuencialmente, o bien, en orden aleatorio distinto en cada iteración. La presentación aleatoria parece más adecuada, ya que si no los últimos casos serían los que más se recordarían.

4. La tasa de aprendizaje. Como se vió en el perceptrón, esta tasa es un valor crítico, ya que para una determinada red y una tasa de aprendizaje infinitesimal, se pueden encontrar los pesos que logran el error mínimo, pero con el inconveniente de la lentitud. Sin embargo, una tasa de aprendizaje demasiado grande, aunque sea mucho más rápido, puede producir oscilaciones entre soluciones no demasiado buenas. Nuevamente la solución está en la experimentación y comparación de resultados tras un determinado número de iteraciones. Lo deseable sería utilizar la tasa de aprendizaje más grande posible que aún garantice la convergencia a la solución óptima (mínimo global).

5. Convergencia y mínimo local. Un mínimo local es simplemente un mínimo que no es el mínimo global real. Cuando se encuentra un mínimo local, el proceso de entrenamiento parece haber logrado la convergencia, ya que el progreso adicional en la reducción del error es muy pequeño.

Existen varias formas de afrontar este problema, la primera es repetir el aprendizaje con distintos pesos iniciales aleatorios y elegir la mejor solución de entre las obtenidas. Una segunda opción es elegir una tasa de aprendizaje más pequeña, lo que disminuye la probabilidad de caer en un mínimo local, aunque también hará más lenta la convergencia.

Otra alternativa es la inclusión del término impulso comentado en la sección 4.4.3, que acelerará la convergencia y evitará el mínimo local. Por último, al utilizar el gradiente descendente incremental, en realidad se utiliza una superficie de error distinta para cada ejemplo de entrenamiento, siendo la media de éstas la que se utiliza para aproximar el gradiente con respecto al conjunto de entrenamiento completo. Cada una de estas superficies de error normalmente tendrán mínimos locales, haciendo menos probable que el proceso quede atrapado en ninguno de ellos.

Aunque no asegura la convergencia al mínimo global, el algoritmo de retropropagación resulta muy efectivo en la práctica y el problema del mínimo local no es tan grave como podría temerse. Ello se debe principalmente a dos motivos:

En primer lugar, las redes con un gran número de pesos se corresponden con superficies de error en espacios de muchas dimensiones (una por cada peso). Cuando el gradiente descendente cae en un mínimo local con respecto a uno de esos pesos, no necesariamente tiene que ser un mínimo local con respecto al resto de pesos. De hecho, cuantos más pesos haya en la red, más dimensiones habrá que puedan proporcionar "rutas de escape" para que el gradiente descendente salga del mínimo local con respecto a ese peso.

Por otro lado, si los pesos inicialmente se igualan a valores cercanos a cero, durante los primeros pasos del gradiente descendente la red representará una función muy suavizada que es casi lineal. Sólo cuando los pesos hayan tenido tiempo de crecer alcanzará un punto donde puedan representar funciones muy distintas a la lineal. Se puede esperar que la mayoría de los mínimos locales estén en la región del espacio de pesos que representa estas funciones más complejas. Es deseable que, llegado el momento en que los pesos alcancen este punto, se hayan acercado ya suficientemente al mínimo global de tal forma que incluso los mínimos locales en esta región sean aceptables.

6. Condición de parada. A la hora de detener el proceso existen varias alternativas, una de ellas es fijar un número máximo de iteraciones y, si el algoritmo no converge antes, se detiene el proceso tal y como esté. Otra alternativa es fijar un error umbral, bien en el conjunto de entrenamiento, o bien, en un conjunto de validación separado. Al igual que en otros métodos de clasificación, el criterio de parada es importante, ya que muy pocas iteraciones puede que no reduzcan el error lo suficiente, mientras que excesivas pueden llevar a sobreajustar los ejemplos de entrenamiento. Otra alternativa es evaluar la reducción del error conseguida después de un número fijo de iteraciones. Si, por ejemplo, el error medio de las últimas 500 iteraciones no es mejor

que el de las 500 anteriores, se puede pensar que no se está progresando y detener el proceso de entrenamiento.

7. Tamaño de la red. Para un conjunto de entrenamiento dado habrá que decidir la estructura de la red que se va a utilizar, el número de capas ocultas y el número de unidades ocultas y de salida. Obviamente esta elección es muy importante, ya que condiciona todas las decisiones anteriores. Como en todos los métodos de aprendizaje, de entre varias redes neuronales que proporcionan los mismos resultados se preferirá la más sencilla, ya que probablemente se comporte mejor ante nuevos casos, pues todo lo referente al sobreajuste comentado hasta ahora es aplicable también al caso de las redes. Además de los conjuntos de entrenamiento y prueba, sería deseable disponer de un tercer conjunto, que haya permanecido totalmente al margen en el entrenamiento de la red, para poder observar realmente el comportamiento de la red definitiva ante nuevos casos.

Si se tiene una red con una única capa de unidades ocultas, y se va aumentando el número de unidades en la capa oculta, desde ninguna a varias y después a muchas, se estará pasando de un ajuste demasiado bajo a un sobreajuste.

Existen algunas reglas sencillas para la selección del número de unidades ocultas, como por ejemplo la que dice que debe haber al menos diez ejemplos por cada peso en la red. Sin embargo, resulta más apropiado utilizar alguna de las técnicas de remuestreo que se vieron en el segundo capítulo, para determinar la complejidad necesaria, de tal forma que se elegirá el número de unidades ocultas que minimice la estimación del error por remuestreo. El principal inconveniente es que, dada la lentitud del algoritmo de retropropagación, el remuestreo multiplicará este efecto y puede llevar a consumir demasiado tiempo.

Además, el análisis se complica si se generaliza al caso de más de una capa oculta, ya que hay que controlar dos variables, por un lado el número de capas y, por otro, el número de unidades por capa.

En la práctica, antes de aplicar alguna de las técnicas de remuestreo, deben seleccionarse los parámetros de entrenamiento que se han visto en esta sección, utilizando el conjunto de entrenamiento completo. Una vez que se han establecido las condiciones de entrenamiento satisfactorias, se aplica alguna de las técnicas de remuestreo.

4.6. ÁRBOLES DE CLASIFICACIÓN

Actualmente es uno de los métodos de aprendizaje inductivo supervisado no paramétrico más utilizado. Este método se caracteriza por la sencillez de su representación y de su forma de actuar. En realidad, lo que realiza es una partición recursiva del espacio multidimensional que constituyen el conjunto de características que describen el problema, el *espacio muestral*.

Un árbol de clasificación se representa gráficamente mediante nodos y ramas. Cada nodo simboliza una cuestión o decisión sobre una de las características de los ejemplos. El nodo inicial se suele llamar nodo raíz. De cada nodo puede salir dos o más ramas, dependiendo de que la respuesta a la cuestión planteada sea binaria o no. Finalmente, se alcanzan los nodos terminales u hojas y se toma una decisión sobre la clase a asignar.

Cuando se presente un nuevo ejemplo o patrón al árbol, éste lo filtrará a lo largo de los test que contienen los nodos. Cada test tiene salidas mutuamente excluyentes y exhaustivas, lo que significa que los ejemplos que se asignen a una de las salidas, no se pueden asignar a otra y además todos los ejemplos se asignarán a una de las salidas. Es decir, ningún ejemplo se asignará a dos salidas de un mismo test, pero tampoco habrá ningún ejemplo que no se asigne a ninguna salida.

Los árboles de decisión pueden trabajar tanto con variables continuas como con variables categóricas, de dos o más categorías. Si sólo se utilizan variables categóricas de dos modalidades, se estará ante el caso particular de un árbol binario. Weiss (1991, pág. 117), utiliza un sencillo ejemplo de un árbol de clasificación binario, donde las

clases son el posible crecimiento en el mercado de acciones o, por el contrario, la bajada en el mismo. Los nodos 1 y 2 plantean, respectivamente, las preguntas de si van a subir o no los tipos de interés en el primer caso y los beneficios de las empresas en el segundo. Por otro lado, los tres nodos terminales u hojas recogen tres grupos mutuamente excluyentes de observaciones. Aquellas en las que los tipos de interés no van a subir, que predice una subida de las acciones. Aquellas en las que los tipos de interés van a subir y además no suben los beneficios de las empresas, que implicaría una bajada de las acciones y, por último, el grupo de los ejemplos donde se incrementan los tipos de interés, pero también se incrementan los beneficios de las empresas, por lo que se puede predecir una subida en el precio de las acciones.

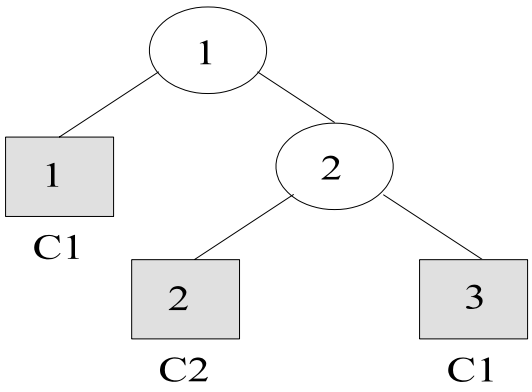


Figura 4.5. Árbol de clasificación binario.

De manera más formal, se puede decir que un árbol de clasificación, *T*, representa una partición recursiva del espacio muestral, *P*, realizada en base a un conjunto de ejemplos, *S*. Además, existen las siguientes relaciones entre los nodos de *T*, las regiones en *P* y los conjuntos de *S*:

1. Cada nodo de T tiene asociado un subconjunto de ejemplos de *S*. En el caso del nodo inicial o raíz tiene asignado el conjunto S completo.

$$\bigcup_{\forall t \in \tilde{T}} R_t = P \qquad \textbf{(4.28)}$$

2. Cada hoja, *t*, tiene asociada una región, R_t, en *P*. Así, si $\tilde{T}$ es el conjunto de hojas del árbol *T*:

que significa que los subconjuntos de ejemplos asignados a los nodos hoja constituyen una partición de *P*. Es decir, ningún elemento de *P* se quedará sin un nodo hoja al cual ser asignado.

3. Cada nodo interior tiene asociada una región en *P*, que es la unión de las regiones asociadas a los nodos hoja del subárbol cuya raíz es él.

4. La unión de los conjuntos de ejemplos asignados a los nodos de un mismo nivel da como resultado el conjunto inicial.

Hay varios aspectos en los que los árboles de clasificación pueden diferenciarse:

1. Las cuestiones pueden ser multivariantes o univariantes, según se evalúen varios atributos de la observación al mismo tiempo o únicamente un atributo.

2. Los nodos internos pueden tener dos salidas o más de dos. Como se ha dicho, si todos los test tienen dos salidas, se tiene un árbol de decisión binario.

3. Los atributos o variables que caracterizan a los ejemplos pueden ser continuos o categóricos.

4. Se pueden tener dos clases o más de dos. Si se tienen dos clases y atributos binarios, el árbol implementa una función Booleana y se llama árbol de clasificación booleano.

5. En el criterio de parada, es decir, cuándo se detiene el crecimiento del árbol.

6. En el criterio de poda, cuando se tiene un árbol demasiado grande y se pretende simplificarlo, existen diferentes formas de hacerlo, las más utilizadas son el criterio de coste-complejidad y la conversión en reglas.

Las principales diferencias entre los métodos de construcción de árboles, vendrán determinadas por estas dos últimas cuestiones, es decir, el criterio de parada y el criterio de poda que se estudiarán más adelante.

4.7. CONSTRUCCIÓN DEL ÁRBOL DE CLASIFICACIÓN

A la hora de construir un árbol de clasificación para un conjunto de entrenamiento concreto son muchas las posibilidades que existen, por lo que es inabordable examinarlas una a una y, por tanto, se hace necesario el establecimiento de una serie de mecanismos para buscar alguno de una forma óptima.

La construcción del árbol se realiza durante la fase de aprendizaje, que puede esquematizarse en los siguientes pasos que se repiten recursivamente:

1. Cada nodo se parte en función de alguna prueba, que normalmente se plantea sobre el valor de alguna de las características que describen los ejemplos. En el caso binario, (verdadero, falso) los ejemplos que cumplen la condición se asignarán a uno de los nodos hijos y los restantes, al otro[8]. Salvo el nodo raíz, cuando se parte un nodo éste pasa a ser un nodo interno o intermedio.

2. La condición de parada detiene el proceso de partición de nodos, cuando un nodo cumple esta condición se dice que es un nodo terminal u hoja. Los ejemplos pertenecientes a un nodo hoja tendrán cierta homogeneidad, por lo que al nodo se le asigna una etiqueta con la clase mayoritaria, si todos los ejemplos del nodo hoja son de la misma clase, se dice que es un nodo puro.

[8] Normalmente los ejemplos que cumplen la condición se asignan a la rama izquierda, mientras que el resto se asignan a la rama derecha.

Este es el esquema general que siguen todos los procedimientos basados en árboles, a la hora de llevarlo a la práctica hay que concretar muchos aspectos, como ¿de qué forma se hacen las particiones y se selecciona la mejor de entre todas las posibilidades?, ¿cuándo hay que detener la partición de los nodos y declararlos como nodos hoja?¿qué etiqueta se le debe asignar al nodo hoja?

4.8. LA REGLA DE CORTE O DIVISIÓN[9]

La partición que se haga separará al conjunto de ejemplos en subconjuntos disjuntos. El objetivo es conseguir subconjuntos más homogéneos, en términos de clase, que el conjunto de partida, es decir, que éstos tengan una mayor pureza que el conjunto inicial. Para cuantificar la homogeneidad se establecerá una medida de impureza que, por un lado, ayudará a elegir la mejor partición de entre las posibles y, por otro, puede utilizarse para detener el crecimiento del árbol cuando esta medida alcance un umbral previamente fijado.

A la hora de formular las preguntas que darán lugar a las particiones, el método más sencillo y a la vez más utilizado es el de plantear sólo preguntas de tipo binario.

Este es por ejemplo el caso del algoritmo CART [10], propuesto por Breiman en 1984, que representa por *Q* al conjunto de posibles preguntas binarias del tipo:

$$\text{¿}\, x \in A\text{?} \quad ; \; A \subset P \quad ; \textit{siendo } P \textit{ el espacio de representación} \qquad \textbf{(4.29)}$$

El conjunto *Q* genera un conjunto de particiones *S* en cada nodo *t*, y cada nodo *t* tendrá dos nodos hijos t_L (el izquierdo) y t_R (el derecho), por convención:

A Los casos de *t* que cumplen $x \in A$ se asignan a t_L ; $t_L = t \cap A$

[9] Estas reglas en Inglés se conocen como splitting rules.

[10] CART, es el acrónimo de Classification And Regression Trees, o árboles de clasificación y regresión.

A Los casos de t que no cumplen x OA se asignan a t_R ; $t_R = t$ 1 $\bar{A}$

Cada pregunta se plantea sobre un único atributo, si el atributo es categórico y dicotómico, no hay que hacer ninguna transformación, ya que la pregunta dará lugar a una partición dicotómica. El problema se plantea cuando el atributo tiene más de dos categorías, o es continuo.

En el primer caso, cuando el atributo tiene más de dos categorías, la pregunta se planteará dividiendo el conjunto de posibles categorías del atributo en dos subconjuntos, de tal forma que la partición resultante sigue siendo binaria. Por ejemplo, si el atributo X_i puede tomar los valores {alto, bajo, medio}, las preguntas posibles son ¿X_i Oalto?; ¿X_i Obajo?; ¿X_i Omedio?.

Cuando la característica es de tipo continuo se establece un valor que divide a los ejemplos en dos, los que toman un valor menor o igual por un lado, y los que toman un valor superior a él por otro. En teoría se puede fijar cualquier valor, pero para simplificar, en general, se utiliza el punto medio entre cada dos valores consecutivos de los que toma la variable en los ejemplos de entrenamiento. Por ejemplo, si X_i es una variable continua y toma los valores 5, 10 y 15, las preguntas que se utilizarían serían, ¿ X_i # 7,5? y ¿ X_i # 12,5?.

4.8.1. Criterios de corte

De entre todas las posibles preguntas y particiones que se pueden hacer, habrá que elegir aquellas que lleven a obtener un mejor resultado, es decir, que obtenga un mayor incremento en la homogeneidad o pureza de los nodos hijos con respecto al nodo padre. Para ello, hay que establecer una medida de impureza que, según Breiman et al (1984, pág. 24), debería ser cero si el nodo es puro y máxima cuando todas las clases tengan las misma proporción de casos. Entre los criterios más utilizados se encuentran la medida de Entropía y la basada en el Indice de Gini.

La medida de Entropía en un nodo *t*, se calcula como

$$i(t) = -\sum_{j=1}^{q} p(j/t)\, Ln\,(p(j/t)) \qquad \textbf{(4.30)}$$

donde se asume que 0ALn(0) = 0 y se estima *p(j/t)* como la proporción de ejemplos de la clase *j* en el nodo *t*, es decir:

$$p(j/t) = \frac{N_j(t)}{N(t)} \qquad \textbf{(4.31)}$$

siendo *N(t)* y $N_j(t)$, el número de ejemplos total del nodo *t* y el número de ejemplos de la clase *j* en *t* respectivamente.

El Indice de Gini mide la concentración u homogeneidad de las clases en un nodo y se calcula como

$$i(t) = -\sum_{\substack{i,j=1 \\ i\neq j}}^{q} p(i/t)\, p(j/t) = 1 - \sum_{j=1}^{q} p(j/t)^2 \qquad \textbf{(4.32)}$$

La elección entre ambos criterios, según Breiman et al (1984), depende del problema, aunque también apuntan que el clasificador generado no es muy sensible a esta elección, según les demuestra la experiencia.

Se puede utilizar el siguiente ejemplo para ver el cálculo de ambas medidas, para ello se parte de dos nodos t_1 y t_2, que están compuestos por ejemplos de tres clases distintas en las siguientes cantidades {5, 10, 25}y {2, 3, 35} respectivamente.

La entropía en cada caso será

$$i(t_1) = -\sum_{j=1}^{3} p(j/t_2)\, Ln\,(p(j/t_2) = -\left(\frac{5}{40} Ln(5/40) + \frac{10}{40} Ln(10/40) + \frac{25}{40} Ln(25/40)\right) = 0{,}9$$

$$i(t_2) = -\sum_{j=1}^{3} p(j/t_2)\, Ln\,(p(j/t_2) = -\left(\frac{2}{40} Ln(2/40) + \frac{3}{40} Ln(3/40) + \frac{35}{40} Ln(35/40)\right) = 0{,}46$$

mientras que, si se calcula la impureza mediante el Indice de Gini, se tiene

$$i(t_1) = 1 - \sum_{j=1}^{3} p(j/t_1)^2 = 1 - \left((5/40)^2 + (10/40)^2 + (25/40)^2\right) = 1 - \frac{750}{1600} = 0{,}531$$

$$i(t_2) = 1 - \sum_{j=1}^{3} p(j/t_2)^2 = 1 - \left((2/40)^2 + (3/40)^2 + (35/40)^2\right) = 0{,}2263$$

por lo tanto, en ambos casos el nodo t_2 resulta más homogéneo o puro que el nodo t_1 .

Para el caso particular de sólo dos clases, las proporciones serán p y 1-p, y es sencillo comprobar que la máxima dispersión se alcanza cuando ambas clases tienen la misma probabilidad, es decir p=0,5

$$i_E(t) = -\sum_{j=1}^{2} p(j/t)\, Ln\,(p(j/t)) = -\left(0{,}5\, Ln(0{,}5) + 0{,}5\, Ln(0{,}5)\right) = 0{,}693$$

$$i_{IG}(t) = 1 - \sum_{j=1}^{2} p(j/t)^2) = 1 - \left((0{,}5)^2 + (0{,}5)^2\right) = 0{,}5$$

Además, se puede representar gráficamente cómo varían al entropía y el índice de Gini, para el caso dicotómico, cuando p lo hace entre 0 y 1.

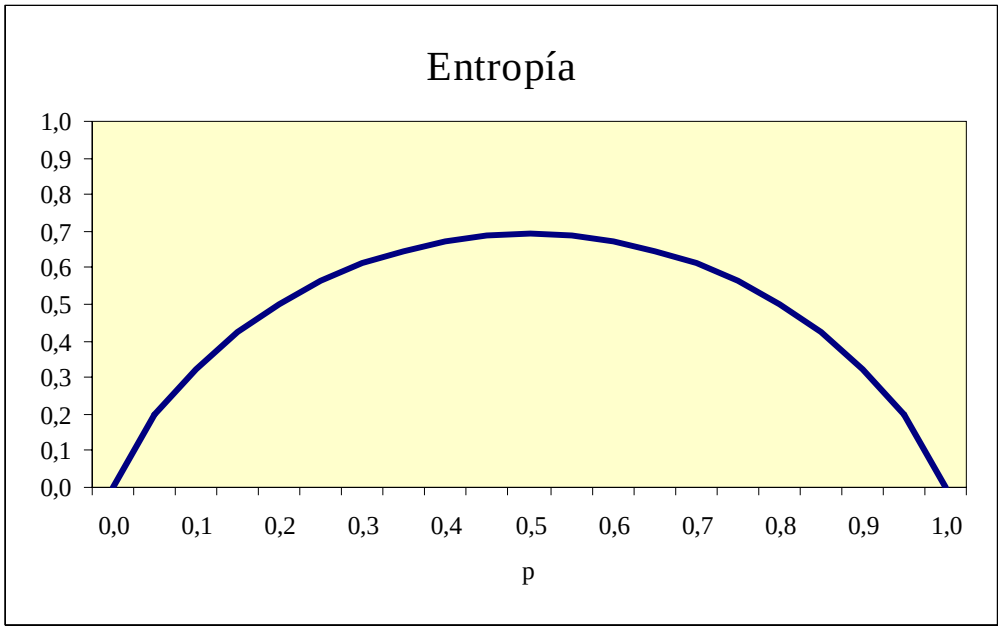


Figura 4.6 Comportamiento de la Entropía ante los valores de p en el caso binario.

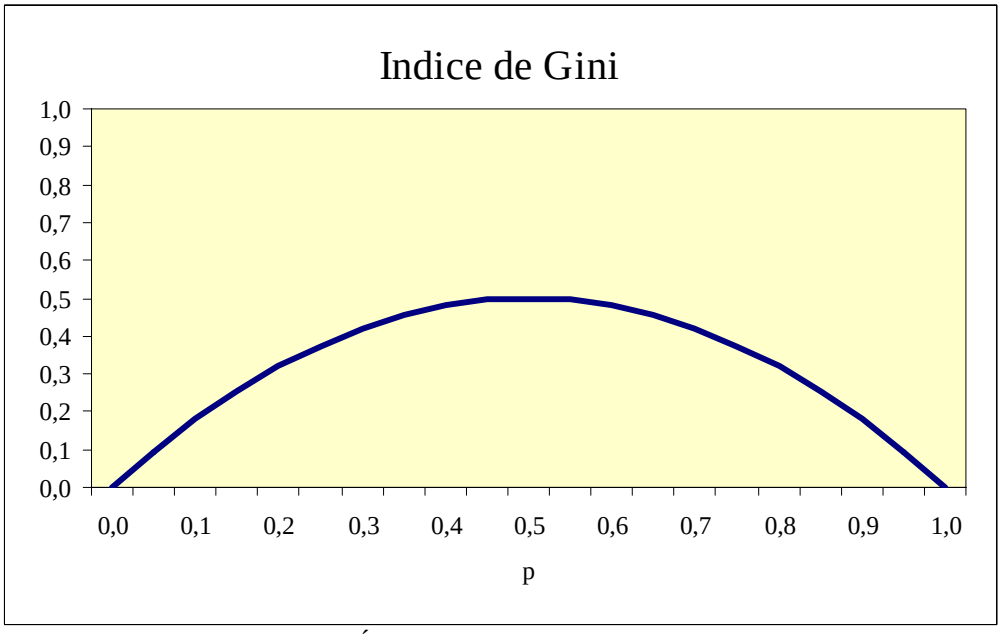


Figura 4.7. Comportamiento del Índice de Gini ante los valores de p en el caso binario.

4.8.2. Bondad de una partición

Cualquiera que sea la medida de impureza que se utilice para medir la bondad de una partición, lo que se hace es comparar la impureza del nodo que se va a dividir, con la impureza de los nodos hijos. Así, para una partición S del atributo A, la bondad se calcula como la ganancia en información, o decrecimiento en impureza, que se consigue separando los ejemplos de acuerdo al atributo. En concreto, la ganancia de información, $\Delta i(S, A)$ de un atributo A respecto a un conjunto de ejemplos S, se calcula como:

$$\Delta i(S, A) = i(S) - \sum_{v=1}^{k} \frac{N_v}{N} i(S_v) \qquad \textbf{(4.33)}$$

donde k es el número de posibles valores del atributo A y N_v/N es la proporción de ejemplos de S que toman el valor v en el atributo A y que, por tanto, forman parte del subconjunto S_v. El segundo término de la ecuación 4.33 es el valor esperado de la impureza después de dividir S utilizando el atributo A, que se calcula sencillamente como la suma de la impureza de cada subconjunto S_v, ponderada por la proporción de ejemplos de S que pertenece a S_v. Por tanto, la ganancia de información es la reducción esperada en la impureza al conocer el valor del atributo A, o bien, la información que el atributo A proporciona sobre la clasificación deseada.

Para el caso particular de divisiones binarias, la ganancia de información en el nodo t será:

$$\Delta i(t) = i(t) - p_L\, i(t_L) - p_R\, i(t_R) \qquad \textbf{(4.34)}$$

De entre todas las posibles particiones, se elegirá aquella que maximice la ganancia de información o la reducción de la impureza. Es decir, primero se elige para cada atributo A_i la partición que maximice la ganancia de información y luego, una vez que se tiene para cada atributo la partición óptima, se elige aquel atributo que suponga una mayor ganancia de información.

Continuando con el ejemplo anterior, si se divide el nodo t_1, que estaba compuesto por {5, 10, 25}, en dos nodos t_{1L}, {2, 8, 2} y t_{1R}, {3, 2, 23} utilizando la entropía, la ganancia de información se calculará como:

$$i(t_{1L}) = -\left(\frac{2}{12} Ln(2/12) + \frac{8}{12} Ln(8/12) + \frac{2}{12} Ln(2/12) \right) = 0{,}868$$

$$i(t_{1R}) = -\left(\frac{3}{28} Ln(3/28) + \frac{2}{28} Ln(2/28) + \frac{23}{28} Ln(23/28) \right) = 0{,}35$$

$$\Delta i(t_1) = i(t_1) - p_L\, i(t_1 L) - p_R\, i(t_1 R) = 0{,}9 - 12/40 \cdot 0{,}868 - 28/40 \cdot 0{,}35 = 0{,}395$$

4.8.3. Impureza de un árbol

A efectos de comparación entre diversos árboles que se pueden construir sobre un mismo conjunto de datos, es interesante conocer la impureza media de un determinado árbol *T*, cuyo conjunto de hojas se puede llamar $\tilde{T}$, que se calculará como:

$$I(T) = \sum_{t \in \tilde{T}} i(t)\, p(t) \qquad \textbf{(4.35)}$$

donde *p(t)* es la probabilidad de que un ejemplo alcance el nodo hoja *t*. Es decir, la impureza de un árbol la determinan sus hojas o nodos terminales.

Breiman et al (1984) concluyen que seleccionar en todos los casos las particiones que maximizan la ganancia de información, es equivalente a seleccionar las particiones que minimizan la impureza global *I(T)*. Es decir, si en cada nodo se elige la mejor partición, se llegará a la solución óptima considerando el árbol final. Cada vez que se divide un nodo *t*, la reducción de la impureza media global es igual a *p(t)* por la ganancia de información obtenida con la partición, por lo que cuando se maximiza la reducción de impureza en *t* se está minimizando, al mismo tiempo, la impureza media del árbol.

Obviamente, si se tienen dos árboles T_1 y T_2 del mismo tamaño, construidos a partir de un mismo conjunto de observaciones y se conoce la impureza de ambos $I(T_1)$ e $I(T_2)$ se preferirá T_1 si $I(T_1) < I(T_2)$.

4.9. CRITERIO DE PARADA

El proceso de división o partición de los nodos deberá detenerse en algún momento, ahora bien, existen varias posibilidades y la elección del criterio de parada que optimice el árbol a obtener no es sencilla:

1. Continuar dividiendo los nodos, hasta que todos los nodos hoja sean puros. Es decir, hasta obtener subconjuntos donde todos los ejemplos sean de la misma clase. Esto puede llevar a nodos hoja con subconjuntos demasiado pequeños, por lo que se puede plantear un segundo criterio.

2. Detener la división de los nodos cuando sean puros, o cuando su tamaño sea inferior a un determinado umbral. El valor umbral dependerá del tamaño inicial del conjunto de entrenamiento ya que, si por ejemplo se parte de un conjunto de mil ejemplos, un nodo con menos de diez casos puede considerarse pequeño mientras que, si el conjunto inicial es de 50 ejemplos, un nodo con igual número de ejemplos no será demasiado pequeño. También habrá que tener en cuenta el número de ejemplos de la clase minoritaria.

3. Un tercer criterio puede ser detener el crecimiento del árbol cuando los nodos hoja superen un determinado nivel de pureza, considerada en términos de la proporción de la clase mayoritaria. Por ejemplo, se puede exigir que la clase mayoritaria supere un 70% para declarar el nodo como nodo hoja y detener la división. Este porcentaje puede fijarse más o menos alto, en función de lo exigente que se quiera ser y de un equilibrio entre la impureza media del árbol y el tamaño del mismo.

4. También se puede utilizar la ganancia de información o reducción de impureza como criterio para detener el crecimiento del árbol. En este caso también se fija un umbral **$**$0, de tal forma que el nodo t será un nodo hoja si:

$$\max_{S} \Delta i(S,t) \leq \beta \qquad \textbf{(4.36)}$$

Es decir, si ninguna de las posibles particiones, *S*, del nodo *t* consigue una reducción de la impureza superior al umbral **$**, se dirá que el nodo *t* es un nodo hoja.

En este caso el umbral **$** resulta un factor crítico ya que, si es demasiado bajo, permitirá que el árbol se desarrolle o crezca mucho, porque será fácil encontrar una partición que consiga una ganancia de información superior a **$**. En cambio, si **$** es demasiado alto, será difícil encontrar una partición cuya reducción de impureza supere el umbral y, por tanto, la construcción del árbol se detendrá prematuramente. Además, es posible que, después de particiones con ganancias de información pequeñas, se obtengan otras particiones que logren una mayor reducción de impureza.

5. Una última razón para detener el crecimiento del árbol nada deseable es agotar la información disponible, cuando se tienen pocas características que describan cada observación puede ser que, después de utilizarlas todas, aún no se hayan encontrado nodos hoja que cumplan ninguno de los criterios anteriores y, sin embargo, será obligatorio detener el crecimiento del árbol.

En realidad, la estrategia que se suele utilizar es construir un árbol muy grande, sin detener su crecimiento, para después proceder a su poda desde las hojas hasta la raíz (sentido inverso al de su construcción), obteniendo un conjunto decreciente de árboles.

Una vez declarado un nodo como hoja, hay que decidir qué clase se asignará a los ejemplos que alcancen ese nodo, tras recorrer el árbol desde la raíz hasta esa hoja. Si todos los ejemplos de entrenamiento de ese nodo pertenecen a una misma clase *C*, entonces obviamente esa hoja se etiqueta con la clase *C*. En el caso de que haya

ejemplos de varias clases, se asignará la clase mayoritaria dentro de esa hoja *t*. Algebraicamente se tiene que:

$$C(t) = C \quad si \quad p(C/t) = \max_{j = 1,2,...q} p(j/t) \tag{4.37}$$

donde como se ha visto en la ecuación 4.31 las probabilidades *p(j/t)* se estiman mediante la proporción de ejemplos de la clase j en el nodo *t*. En caso de empate entre dos o más clases, se elige una de ellas al azar.

4.10. SOBREAJUSTE EN LOS ÁRBOLES DE CLASIFICACIÓN

Cuando se desarrolla un árbol hasta alcanzar hojas puras o muy pequeñas, en las últimas divisiones la selección de la partición más adecuada, se hace en base a un conjunto de ejemplos muy pequeño, por lo que se corre el riesgo de *sobreajuste o sobreaprendizaje*. Esto puede deberse bien a que el conjunto de entrenamiento contiene errores aleatorios o ruido, o bien, a regularidades que son frutos de una coincidencia pero no de una verdadera relación entre la partición y la clasificación a determinar.

Los diversos métodos que existen para evitar el sobreajuste al construir un árbol de clasificación se pueden agrupar en dos grandes grupos:

1. Los métodos que detienen el crecimiento del árbol antes de que los nodos hoja sean puros o demasiado pequeños.

2. Los métodos que desarrollan el árbol hasta que los nodos hoja sean puros o muy pequeños, para después aplicar algún mecanismo de poda al árbol completo.

A priori el primer método puede parecer más rápido y mejor, sin embargo, Breiman et al (1984) defienden la poda a posteriori frente a la interrupción del crecimiento. Según ellos resulta más eficiente la poda, ya que permite que uno o varios subárboles

de un nodo permanezcan y el resto desaparezcan mientras que, si se interrumpe el crecimiento, todas las ramas son eliminadas.

En Weiss y Kulikowski (1991) se propone utilizar técnicas de validación cruzada para estimar el error real del árbol sobre el total de la población, ya que el error en reescritura es un estimador optimistamente sesgado y que puede seguir decreciendo hasta llegar a cero cuando todas las hojas son puras. En cambio el error real, deja en un determinado momento de descender, se mantiene más o menos constante para un número de nodos hoja ligeramente superior y, seguidamente, empieza a crecer conforme el árbol aumenta de tamaño, ver figura 4.8. Lo que se propone es detener el crecimiento del árbol cuando el error real alcance su mínimo. Es decir, si una partición no disminuye el error real, entonces no se realiza esa partición y el nodo se declara terminal. El árbol que resulta es un subárbol del árbol completo y su error no será menor que la mitad del error real del árbol completo. Es decir, como mucho se puede mejorar el error real del árbol en un cincuenta por ciento.

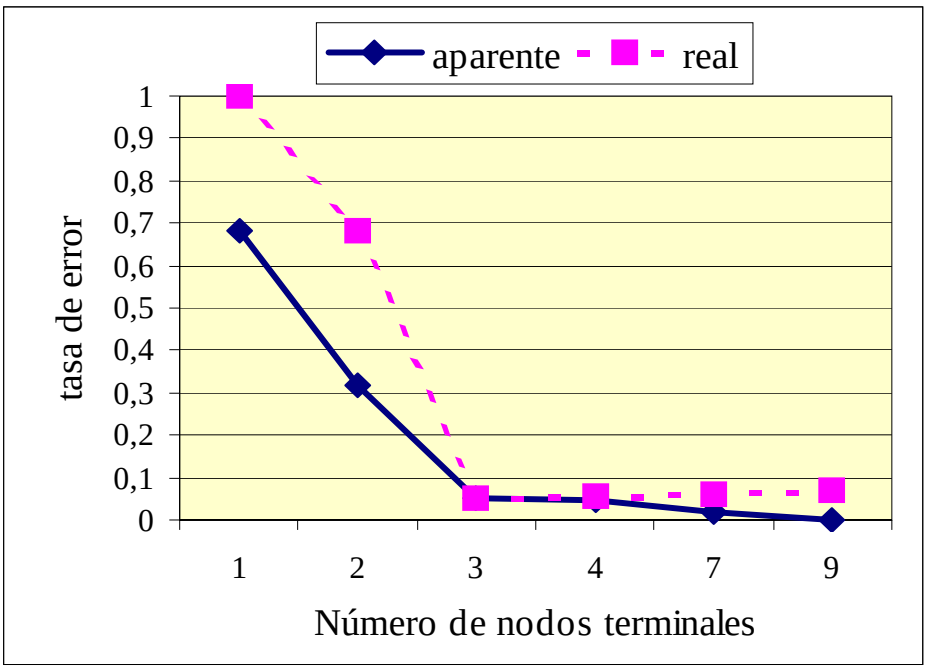


Figura 4.8. Evolución de las tasas de error en función del tamaño del árbol.

Para determinar el tamaño correcto del árbol, se pueden utilizar varios criterios:

1. Utilizar un conjunto distinto al de entrenamiento para evaluar la eficacia de la poda de nodos del árbol.

2. Utilizar todos los ejemplos disponibles para el entrenamiento y aplicar una prueba estadística para decidir cuándo la poda de un nodo puede producir una mejora en el conjunto de entrenamiento. Quinlan (1986) utiliza una prueba $\mathbf{P}^2$ para estimar si la división de un nodo mejora el comportamiento sobre la población total de ejemplo, o sólo sobre la muestra del conjunto de entrenamiento.

3. Utilizar una medida explícita de la complejidad para codificar los ejemplos de entrenamiento y el árbol de clasificación, deteniendo el crecimiento del árbol cuando se minimiza el tamaño de la codificación. Estos métodos se basan en el principio de longitud mínima de la descripción[11].

4.11. TÉCNICAS DE PODADO[12]

Como se ha comentado, resulta más eficiente desarrollar el árbol por completo y podarlo posteriormente, para ello se pueden emplear diversas estrategias cuyo procedimiento general será:

1. Desarrollar el árbol hasta que todas las hojas sean puras o tengan un tamaño muy pequeño. Se obtendrá así un árbol demasiado grande que se representa por T (árbol completo).

2. Una vez obtenido el árbol completo T, se procede a podarlo logrando una secuencia decreciente y anidada de árboles. Si T_1 es el resultado de podar T, se dice que T_1 es un subárbol podado de T y se representa como $T_1 \leftarrow T$, de tal forma que se puede

[11] Para una descripción más detallada ver Quinlan y Rivest (1989)

[12] Las técnicas de podado se conocen en Inglés como pruning rules.

representar la secuencia completa de árboles podados y anidados como, $T_{max} \succ T_1 \succ T_2 \succ T_3 \succ \ldots \succ t_1$ donde t_1 es un árbol con un único nodo.

Se trata, por tanto, de elegir uno de esos árboles y para ello se asignará una medida de error a cada uno de esos árboles seleccionando, finalmente, el árbol que tenga un menor error. Se supone que árboles más pequeños obtendrán menores errores al evaluarlos sobre nuevas observaciones, ya que no estarán sobreajustados y tendrán una mayor capacidad de generalización, frente a los árboles más complejos, que tenderán a estar sobreajustados a los ejemplos de entrenamiento y lograrán peores resultados ante nuevas observaciones.

4.11.1. Podado por mínimo coste-complejidad[13]

Este método de poda por coste-complejidad, también se conoce como poda por la rama más débil. A partir del árbol completo se poda hacia atrás, para evitar tener que considerar todos los posibles subárboles podados, se consideran sólo los subárboles obtenidos podando la rama más débil en el árbol. La rama más débil del árbol se determina considerando al mismo tiempo el error cometido y el tamaño del árbol que se representará por $|\tilde{T}_t|$. En concreto, la rama más débil será aquel subárbol enraizado en un nodo no terminal *t*, que puede ser borrado con el menor incremento en el error por nodo podado. En cada nodo *t*, se cuantifica el efecto de podar el subárbol enraizado en *t* mediante *g(t)*, que se calcula como:

$$g(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t|} \qquad \textbf{(4.38)}$$

Aquel nodo *t* que menos empeore su comportamiento *g(t)*, es podado y el proceso se repite para lo que queda del árbol.

[13] Del Inglés cost-complexity pruning.

Esta estrategia de poda incluye un parámetro que prima los árboles más sencillos frente a los más grandes, es decir, penaliza la complejidad de los árboles. En realidad lo que se busca es un equilibrio entre la precisión de la clasificación del árbol y el tamaño del mismo, de tal forma que se permitirá un incremento pequeño en el error si ello lleva a una gran reducción en el tamaño. Ese factor de penalización supone un umbral que se va incrementando, dando lugar a una secuencia de árboles anidados.

La medida de coste-complejidad del árbol T , que se representará por $R_\alpha(T)$, es una combinación lineal del coste o error del árbol T [14] y su complejidad, ponderada adecuadamente. El error de clasificación de T se representa por $R(T)$, y por complejidad de T se entiendes su tamaño o número de hojas, $|\tilde{T}|$. El parámetro de complejidad α es un valor real ($\alpha \geq 0$), que representa el coste de complejidad por cada hoja.

En concreto la medida de coste-complejidad se calcula como

$$R_\alpha(T)=R(T)+\alpha|\tilde{T}| \qquad \textbf{(4.39)}$$

Un valor grande de α, supondrá un coste alto por cada hoja y favorecerá a los subárboles con menor número de nodos terminales. Por tanto, para cada valor de α, hay que encontrar el árbol $T(\alpha)$, anidado con el árbol completo T que minimice $R_\alpha(T_i)$, es decir, que obtenga

$$R_\alpha(T(\alpha))=\min_{T_i \preceq T}\{R_\alpha(T_i)\} \qquad \textbf{(4.40)}$$

De tal forma que, si se va incrementando progresivamente α, a partir de $\alpha=0$, se generará una secuencia finita y única de subárboles anidados $T_1 \succ T_2 \succ T_3 \succ \ldots \succ t_1$, donde $T_k=T(\alpha_k)$ y $\alpha_1=0$

[14] Aunque en este trabajo se utiliza la estimación del error de clasificación como coste del árbol, también puede utilizarse la entropía o la desviación de la partición.

El procedimiento concreto para obtener la secuencia anterior de árboles, puede encontrarse en Breiman et al (1984), en lineas generales se parte de un árbol con muchas hojas, representado por T_1 (α_1=0). Se busca la rama más débil de T_1 y se poda, creando T_2 cuando α alcanza a α_2. Entonces se busca la rama más débil de T_2 y se poda, creando T_3 cuando α alcanza α_3, y así sucesivamente, hasta llegar al árbol constituido únicamente por el nodo raíz. A medida que crece α, los árboles van siendo cada vez más pequeños, por lo que se obtiene una secuencia decreciente de subárboles anidados.

Se debe elegir el árbol óptimo entre los anteriores, que será aquel que tenga un menor valor para la medida de coste complejidad, es decir, se escogerá T_{opt} tal que

$$\hat{R}(T_{opt}) = \min_{\forall k} R(T_k) \qquad \textbf{(4.41)}$$

En cuanto a las consideraciones sobre la estimación del error de clasificación, véase el apartado 2.6.

La regla 1-SE.

Según Breiman et al (1984), en lugar de elegir el mínimo $R(T_k)$ proponen la regla 1-SE, según la cual se debe elegir aquel árbol de menor tamaño cuyo error no supere el error mínimo más una vez su desviación típica[15]. En concreto la regla 1-SE, será si T_{k0} es el árbol tal que $R(T_{k0}) = \min_k R(T_k)$, entonces el árbol seleccionado será T_{k1}, donde k_1 es el máximo k que satisface [16]:

$$R(T_{k1}) \leq R(T_{k0}) + SE(T_{k0}) \qquad \textbf{(4.42)}$$

[15] La desviación típica del error se conoce como Error Estándar, en inglés Standard Error, de ahí que se represente por SE. Y la regla se represente por 1-SE.

[16] Se elige el máximo k porque será el que dé el árbol más sencillo.

La regla 1-SE consigue por un lado reducir la inestabilidad asociada a la selección del mínimo exacto y, por otro, selecciona el árbol más simple cuya bondad es comparable a la del mínimo $R(T_k)$. Esta regla se comprende fácilmente observando la figura 4.9 que relaciona el error de validación cruzada, la medida de coste-complejidad de un árbol y su número de hojas. Según Breiman et al (1984), la evolución del error de los árboles se caracteriza, en general, por un rápido descenso inicial, que se detiene en una zona plana (valle), más o menos prolongada para, finalmente, ascender cuando el tamaño del árbol es elevado.

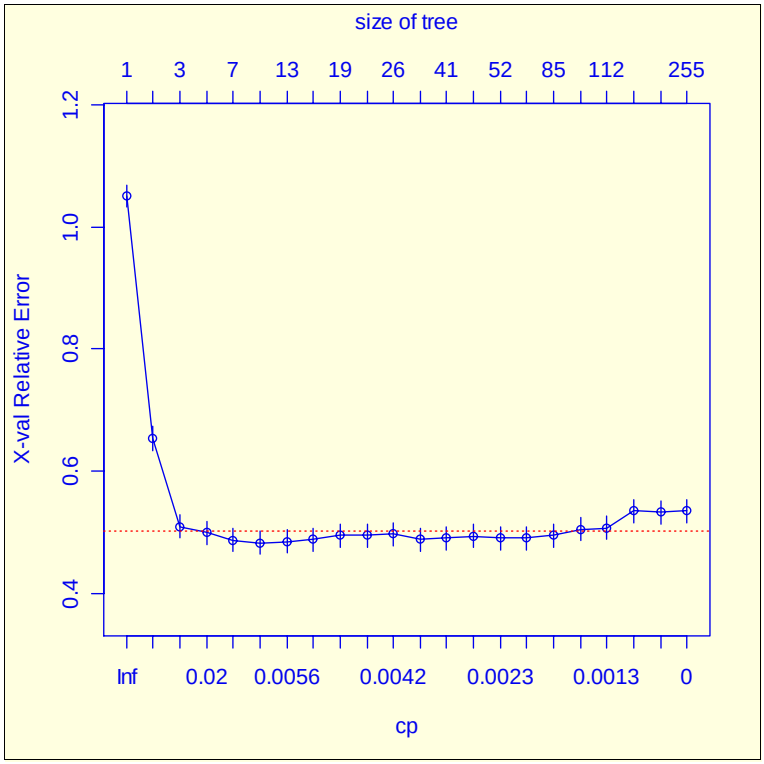


Figura 4.9 Evolución del error de validación cruzada en función del tamaño del árbol y de la medida de coste-complejidad.

4.11.2. Podado de reglas

Este método de poda es utilizado por el sistema C4.5 propuesto en Quinlan (1993)[17], y consiste en convertir el árbol completamente desarrollado en un conjunto de reglas,

[17] El sistema C4.5 nace del desarrollo del sistema ID3 original del mismo autor.

una para cada hoja, y posteriormente simplificar este conjunto de reglas. Los pasos a seguir son los siguientes:

1. En primer lugar, se construye el árbol que mejor se ajuste al conjunto de entrenamiento, sin preocuparse de si existe o no sobreajuste.

2. Se convierte el árbol aprendido en un conjunto de reglas equivalente al árbol, creando una regla para cada camino desde la raíz hasta las hojas, es decir, habrá tantas reglas como nodos hoja.

3. Podar cada regla, eliminando cualquier antecedente de la regla siempre que se consiga mejorar la precisión estimada.

4. Una aportación a este método de poda es introducir un cuarto paso antes del último, en el cual se ordenen las reglas podadas por su precisión estimada y se compruebe la precisión del conjunto de reglas dejando al margen cada vez una de ellas, de menor a mayor precisión. Si la precisión no disminuye, se elimina la regla y se sigue el proceso con el resto. Si la precisión empeora, la regla se añade de nuevo al conjunto y se continúa probando con el resto.

5. Las reglas que no hayan sido eliminadas formarán el conjunto de reglas final, a la hora de aplicarlas se conserva el orden de mayor a menor precisión.

En este método, se genera una regla para cada hoja del árbol completo. Cada variable evaluada a lo largo del camino desde la raíz hasta la hoja se convierte en un antecedente de la regla, y la clase que etiqueta el nodo hoja se convierte en el consecuente de la regla.

A continuación en cada regla se elimina cualquier antecedente, cuya eliminación *no empeore* la precisión estimada de la regla. Para las reglas con dos o más antecedentes, se estima la precisión de la misma si se elimina cada uno de los antecedentes, y se

elimina aquél que produzca un mayor incremento en la precisión. Posteriormente, se repite el proceso para decidir si eliminar un segundo atributo, y así sucesivamente, hasta que no haya ningún antecedente cuya eliminación no empeore la precisión. Ningún antecedente será podado si con ello se empeora la precisión estimada de la regla.

Para estimar la precisión de la regla, además de los métodos comentados en el apartado 2.6, en Quinlan (1993) se propone utilizar una modificación pesimista del error aparente o en reescritura, para compensar el sesgo optimista de esta estimación. Para ello recoge el extremo superior del intervalo de confianza al 95% para la estimación del error, calculando su desviación típica suponiendo una distribución Binomial. Por tanto el valor utilizado para la estimación será el error en reescritura más 1,96 veces la desviación típica estimada[18]. Esta solución facilita mucho los cálculos, disminuyendo el elevado coste computacional de los métodos de remuestreo vistos en el apartado 2.6.

Podar las reglas en lugar de podar el árbol directamente presenta tres ventajas principalmente:

1. Trabajar con reglas permite distinguir entre los diferentes contextos en los que un nodo de decisión es utilizado, ya que el mismo nodo interior pertenece a varias reglas, y será tratado de forma independiente en cada una de ellas mientras que, si se podase el árbol, la decisión sería podar el nodo o no.

2. Se elimina la distinción entre los atributos evaluados cerca de la raíz y aquellos evaluados cerca de las hojas.

3. Mejora la legibilidad. Las reglas suelen ser más fáciles de entender por los humanos.

[18] El valor 1,96 corresponde al valor de la distribución Normal(0,1) para el nivel de confianza del 95%.

4.12. VALORES OMITIDOS [19]

En algunos casos, los datos disponibles incluyen algunos ejemplos con valores omitidos o desconocidos para una o más de las variables o atributos. En estos casos, es común estimar el valor desconocido para el atributo a partir del resto de ejemplos para los que el valor del atributo es conocido.

Si se está en un nodo *t* y uno de los ejemplos incluidos en este nodo tiene un valor desconocido para el atributo *A*, existen varias posibilidades para estimar el valor de *A* en ese ejemplo. La más sencilla es asignarle el valor mayoritario entre los ejemplos de entrenamiento incluidos en *t*, en el caso de que *A* sea categórico; si fuese continuo, se le asignaría la media de los valores que toma *A* en esos ejemplos. Otra posibilidad es asignar el valor más común o la media igual que antes, pero considerando sólo aquellos ejemplos de *t* que son de la misma clase que el ejemplo cuyo valor en *A* se desconoce.

Una tercera alternativa, algo más compleja, es asignar una probabilidad a cada uno de los posibles valores de *A*. Estas probabilidades se estiman a partir de las frecuencias observadas para cada valor de *A* en los ejemplos del nodo *t*. Si por ejemplo la prueba del nodo *t* es dicotómica para el atributo *A* y sus probabilidades son 0,7 y 0,3 respectivamente, un 70% del ejemplo recorrerá la rama izquierda, y un 30% del ejemplo recorrerá la rama derecha. Esta fracción del ejemplo será utilizada para calcular la ganancia de información, y será nuevamente dividido en las siguientes ramas si existe un segundo atributo cuyo valor se desconoce y se establece la prueba sobre ese atributo. El mismo procedimiento puede utilizarse para clasificar nuevas observaciones una vez aprendido el árbol si tienen valores desconocidos, la clase que se asignará será la que tenga mayor probabilidad, si se suman los pesos del ejemplo fraccionado en las distintas hojas. Este procedimiento es el utilizado por el sistema C4.5 de Quinlan (1993).

[19] En inglés Missing values.

SEGUNDA PARTE

CAPÍTULO 5

PROBLEMÁTICA DE LOS CLASIFICADORES INDIVIDUALES

5.1. INTRODUCCIÓN

En la primera parte de este trabajo se han planteado algunos aspectos generales de los problemas de clasificación, como son la selección de variables y la estimación de la precisión de los clasificadores. También se han visto algunos de los métodos estadísticos de clasificación más utilizados. Estos métodos se pueden dividir de acuerdo a los siguientes grupos: técnicas estadísticas tradicionales o clásicas, el método del vecino más próximo, las redes neuronales y, por último, los árboles de clasificación. De ahora en adelante a este tipo de clasificadores se les llamará clasificadores o métodos de clasificación individuales o básicos.

En este capítulo se van a analizar algunos aspectos relacionados con el comportamiento y las propiedades de estos clasificadores individuales. El enfoque se centrará en algunos de los problemas o dificultades que plantea el uso de los

clasificadores individuales, como pueden ser la precisión y la estabilidad de los mismos.

El capítulo se estructura en cuatro etapas. En primer lugar se estudia el "error de generalización" de los clasificadores individuales, es decir, el error que se espera que cometa cuando se aplique en observaciones nuevas, que no estaban presentes en el conjunto donde ha sido entrenado. Este error se desagregará en la suma de tres términos no negativos que son el riesgo de Bayes, el sesgo y la varianza. El primero de ellos recoge el error inherente al conjunto de datos y que ningún método de clasificación puede reducir. El sesgo mide el error persistente del método de clasificación, es decir, el error que se mantendría incluso si se tuviese un conjunto infinito de clasificadores, entrenados de manera independiente. El término correspondiente a la varianza mide el error causado por las fluctuaciones que se producen al generar un clasificador individual. Por tanto, a partir de este enfoque la idea que se desprende es que promediando varios clasificadores se puede reducir el término varianza y, de este modo, disminuir también el error esperado o de generalización.

En segundo lugar, se aborda el tema de la "inestabilidad" de algunos clasificadores, entendiéndose por un clasificador inestable aquel que sufre grandes cambios ante pequeñas modificaciones en el conjunto de entrenamiento. En este sentido los árboles de clasificación y las redes neuronales son considerados como métodos inestables, mientras que el análisis discriminante lineal y el método del vecino más próximo se consideran estables. Sin embargo, cuando el tamaño del conjunto de entrenamiento es demasiado pequeño en comparación con la dimensión de los datos, todos los clasificadores sufren problemas de inestabilidad. Es decir, ante estas circunstancias incluso el discriminante lineal es inestable.

En general, los clasificadores más estables se comportan mejor, mientras que los clasificadores más inestables obtienen peores resultados en la predicción. Por tanto, el estudio de la estabilidad es interesante si se pretende mejorar la precisión de un clasificador. En este sentido, se plantea una medida de inestabilidad, que refleja cómo

afectan al clasificador que se está construyendo los cambios en la composición del conjunto de entrenamiento.

El cuarto apartado analiza algunas razones que explican la superioridad de los métodos de combinación sobre los clasificadores individuales. Así por ejemplo, si se supone que la tasa de error de cada clasificador es inferior al cincuenta por ciento y que los clasificadores individuales son independientes, el error cometido por la combinación de esos clasificadores puede reducirse hasta cero cuando el número de clasificadores básicos se aproxima a infinito. Por otra parte, la combinación ideal consiste en utilizar clasificadores lo más precisos posible y que están en desacuerdo el mayor número de veces, ya que la combinación de clasificadores idénticos no aporta ningún beneficio.

Siguiendo con ese razonamiento, se plantean tres razones para justificar el uso de clasificadores combinados, una estadística, una de computación y una tercera de representación. La primera se basa en la eliminación del riesgo asociado a la elección de un clasificador frente al resto, especialmente en conjuntos de tamaño reducido donde la precisión estimada puede ser similar para varios de ellos. La razón de computación plantea la posibilidad de que los sistemas de clasificación que realizan una búsqueda local puedan quedar atrapados en un óptimo local y verse imposibilitados para alcanzar el óptimo global. Finalmente, la razón de representación aduce la imposibilidad de representar la verdadera función con ninguna de las hipótesis disponibles.

Para acabar este capítulo se recoge un breve comentario sobre la perspectiva bayesiana en la agregación de modelos, que también aboga por esta solución cuando el objetivo no es la selección explícita de un modelo, sino extraer el mejor rendimiento posible al conjunto de datos disponible para la predicción que se desea realizar.

5.2. DESCOMPOSICIÓN DEL ERROR. ESTUDIO DEL SESGO Y DE LA VARIANZA

Para estudiar el error de generalización de un clasificador y analizar su descomposición en varios términos, entre ellos el sesgo y la varianza, se parte como en cualquier problema de clasificación de un conjunto de entrenamiento T, que recoge *n* ejemplos de las *q* clases conocidas. Para cada ejemplo se conocen los valores de *p* atributos o características, que pueden ser numéricos o categóricos, y forman el vector $x = (x_1, x_2, x_3,....,x_p)$ y también su clase *y*. A partir de T se construye un clasificador C(*x*, T), en esta sección se va a estudiar el sesgo, varianza y error de predicción del clasificador C(*x*, T). Para mayor simplicidad, de las diferentes opciones existentes a la hora de elegir la función de pérdida, se elige aquella que se limita a contar los errores cometidos por el clasificador C(*x*, T). Entre las funciones de pérdida alternativas está la suma de cuadrados de los errores (Tibshirani, 1996).

Suponiendo que las observaciones (x_i, y_i) en el conjunto de entrenamiento T son una muestra aleatoria de la verdadera distribución F, normalmente esta distribución se desconoce en la práctica,

$$x_1 , x_2 , ... x_n \overset{i.i.d.}{\sim} F, \tag{5.1}$$

el error de predicción del clasificador C(*x*, T) se define como

$$PE(Y,C)) \equiv E_T E_F (C(x, T) \neq Y) \tag{5.2}$$

donde E_T se refiere a la esperanza sobre el conjunto de entrenamiento T, cuyos miembros son independientes e idénticamente distribuidos (i.i.d.) a partir de F, y E_F representa la esperanza sobre el conjunto de prueba $\tilde{T}$, que también se genera a partir de F. Efron y Tibshirani (1995) diferencian esta tasa de error esperado de la tasa de error condicionada que fija el conjunto de entrenamiento T. Aunque la tasa de error condicionada, puede a menudo ser más interesante, Efron y Tibshirani (1995) muestran que es muy difícil estimarla, por lo que en general se trabaja con la tasa de error no condicionada.

Una vez definido el concepto de error de predicción, se busca ahora la descomposición de este error en varios términos de forma que se facilite su estudio. Antes de abordar la descomposición del error en los problemas de clasificación, se recoge un análisis similar para el caso de los problemas de regresión donde se encuentra más desarrollado.

En los problemas de regresión habitualmente se utiliza como función de pérdida la suma de los cuadrados de los errores, en ese caso, se puede descomponer fácilmente el error de predicción. De este modo, si Y es una variable aleatoria continua

$$Y = C_O(x) + \varepsilon \tag{5.3}$$

donde $E(\varepsilon/x) = 0$. Sea $\sigma^2(x) = Var(Y/x)$, entonces para un estimador $C(x, T)$ con $C_A(x) = E_T[C(x, T)]$, se puede ver que

$$\begin{aligned} PE(Y,C(x,T)) &= E_T E_F[Y - C(x,T)]^2 = E_T E_F[(Y - C_A(x)) - (C(x,T) - C_A(x))]^2 \\ &= E_T E_F[(Y - C_A(x))^2 + (C(x,T) - C_A(x))^2 - 2(Y - C_A(x))(C(x,T) - C_A(x))] = \\ &E_T E_F[Y - C_A(x)]^2 + E_T E_F[C(x,T) - C_A(x)]^2 \end{aligned} \tag{5.4}$$

por otro lado

$$\begin{aligned} E_T E_F[Y - C_A(x)]^2 &= E_F[Y - C_A(x)]^2 = \\ E_F[(C_0(x) - C_A(x)) + \varepsilon]^2 &= E_F[(C_0(x) - C_A(x))^2 + \varepsilon^2 + 2(C_0(x) - C_A(x))\varepsilon] \\ &= E_F[C_0(x) - C_A(x)]^2 + E_F[\varepsilon^2] \end{aligned} \tag{5.5}$$

y si se define

$$\begin{aligned} Sesgo^2(C) &= E_F[C_A(x) - C_O(x)]^2 = E_F[Y - C_A(x)]^2 - E_F[\varepsilon^2] \\ Var(C) &= E_T E_F[C(x,T) - C_A(x)]^2 \end{aligned} \tag{5.6}$$

se obtiene la descomposición

$$PE(Y,C) = E_F[\varepsilon^2] + Sesgo^2(C) + Var(C) \quad \textbf{(5.7)}$$

Por lo tanto, en los problemas de regresión el error de predicción se puede descomponer en la suma de tres términos no negativos. El primero supone un límite inferior en el error de predicción de cualquier método de regresión, y se interpreta como aquella parte de la variabilidad de la variable explicada que no es posible explicar mediante ningún método de regresión, sería algo así como el error intrínseco a un determinado conjunto de datos. La segunda componente, se conoce como sesgo y recoge en qué medida se acerca en promedio el procedimiento de regresión utilizado a la función objetivo, la que conseguiría ese error mínimo. Por último, el tercer término mide cuánto varían las predicciones del procedimiento de regresión utilizado con respecto al promedio.

Una descomposición similar puede obtenerse en clasificación para las medidas de error que generalmente se utilizan. En primer lugar, el clasificador ideal se define por $C^*(x) / E_F(Y/x)$, y se conoce como el clasificador óptimo de Bayes, que se obtiene de la siguiente manera $C^*(x) = \arg\max_{y_j} P(y_j/x)$. Es decir, el clasificador óptimo de Bayes, dada una observación *x*, le asigna aquella clase y_j que tiene una mayor probabilidad de ocurrir condicionada a los valores que ha tomado la observación *x*.

El clasificador agregado o combinado se define como $C_A(x) / E_T C(x, T)$. Para entender lo que este clasificador agregado significa puede verse del siguiente modo. Supóngase que existe un grupo infinito de conjuntos de entrenamiento y se aplica el clasificador $C(x, T)$ a cada uno de ellos, el clasificador agregado $C_A(x)$ sería la media de $C(x, T)$ en *x* sobre este grupo infinito, si la salida de $C(x, T)$ es continua (probabilidades a posteriori), o bien, la clase predicha con mayor frecuencia si el clasificador se limita a asignar una clase. El clasificador combinado reduce el error de $C(x, T)$ agregándolo sobre los conjuntos de entrenamiento obtenidos de F.

Nótese que el clasificador agregado $C_A(x)$ puede variar dependiendo de si C(x, T) devuelve probabilidades a posteriori, o únicamente asigna la clase vencedora. Por ejemplo, si para una observación x, C(x, T) asigna las probabilidades (0.45, 0.55) ó (0.9, 0.1), con probabilidades 2/3 y 1/3. Entonces $C_A(x)$ = (0.6, 0.4) y, por tanto, predice la primera clase. Pero si C(x, T) da exclusivamente la clase, es decir (0, 1) ó (1, 0) con probabilidades 2/3 y 1/3, entonces $C_A(x)$=(1/3, 2/3) y en consecuencia predecirá la segunda clase.

Breiman (1996) utiliza el término agregado y llama a su estimación bootstrap, el clasificador bagging ("bootstrap and aggregating"). Este clasificador se calcula como $\hat{C}_A(x) = E_{\hat{F}}[C(x,T)]$. Bagging imita al clasificador agregado promediando C(x, T) sobre los conjuntos de entrenamiento obtenidos a partir de la estimación de F. Dicha estimación se realiza generando réplicas bootstrap del conjunto de entrenamiento original. En Breiman (1996) se muestra que bagging reduce el error de clasificación al menos un 20% sobre el grupo de problemas utilizado[20]. Además, Breiman obtiene diferencias muy pequeñas cuando bagging trabaja con estimaciones de las probabilidades de las clases, o con la clase directamente. El método bagging se analizará con más detalle en el próximo capítulo.

A partir de las definiciones del clasificador óptimo de Bayes, C*(x), y del clasificador agregado, $C_A(x)$, se pueden definir ahora el sesgo y la varianza para el clasificador C(x, T), que abreviadamente se representa por C, de la siguiente manera :

$$\text{Sesgo}(C) \;/\; \text{PE}(C^*, C_A)$$

[20] En su artículo *"Bagging Predictors"* Breiman utiliza siete conjuntos de la base de datos de la Universidad de California todos ellos reales, excepto uno que es simulado (waveform). Cada conjunto se divide aleatoriamente en un conjunto de entrenamiento y otro de prueba. El árbol individual (CART) se construye de la manera habitual en el conjunto de entrenamiento y para construir el clasificador bagging se realizan cincuenta réplicas bootstrap de este conjunto. Este proceso se repite cien veces para los siete problemas. La comparación entre CART y bagging se realiza utilizando el promedio de los errores a lo largo de las 100 repeticiones.

$$\text{Var (C)} \;/\; \text{PE(C, C}_A\text{)}$$

En base a estas definiciones se desprenden las siguientes propiedades:

1. El sesgo es en realidad un tipo de error cuadrático y, por tanto, nunca es negativo.
2. El clasificador C es insesgado si su versión agregada C_A predice la misma clase que el clasificador óptimo de Bayes, con probabilidad uno sobre las variables de entrada *x*.
3. La varianza nunca es negativa.
4. Si el clasificador C es determinista, es decir, no depende del conjunto de entrenamiento, entonces $C = C_A$ y su varianza es cero.

La segunda propiedad afirma que C(*x*, T) es insesgado en *x* si $C_A(x) = C^*(x)$. Es decir, C(*x*, T) es insesgado en *x* si, en las réplicas de T, C(*x*, T) elige la clase correcta con mayor frecuencia que cualquier otra clase. Un clasificador que es insesgado en *x* no es necesariamente un clasificador infalible. Por ejemplo, supóngase que en un problema dicotómico P(1/*x*)=0,9 ; P(2/*x*)=0,1 y $P_T(C(x, T)=1)=0,6$; $P_T(C(x, T)=2)=0,4$. Entonces C es insesgado en *x*, porque elige la clase correcta con mayor probabilidad que el resto, pero la probabilidad de que C acierte en la clasificación es 0,6*0,9+0,4*0,1=0,58. Mientras que el clasificador C* de Bayes tiene una probabilidad de acierto de 0,9.

La definición de Var (C) / PE(C, C_A) es equivalente a PE(C_A, C), pero de esa forma resulta más natural puesto que permite definir formalmente la versión agregada del clasificador C como

$$C_A(x) = arg\min_{C'} PE(C, C') \qquad \textbf{(5.8)}$$

Tomando como referencia la descomposición del error de predicción obtenida para los problemas de regresión en la ecuación (5.5) y utilizando las definiciones de sesgo y varianza de un clasificador que se acaban de señalar, Tibshirani (1996) plantea la siguiente descomposición del error de predicción de un clasificador.

$$\text{PE(C)} = \text{PE(C*)} + \text{Sesgo (C)} + \text{Var(C)} \qquad \textbf{(5.9)}$$

Para una función objetivo y un tamaño del conjunto de entrenamiento fijos, la formulación convencional de la descomposición divide el error esperado en la suma de tres cantidades no negativas:

"Ruido objetivo" (intrínseco). Esta cantidad es un límite inferior en el error esperado de cualquier algoritmo de aprendizaje. Es el error esperado del clasificador óptimo de Bayes y se conoce en ocasiones como riesgo de Bayes.

"Sesgo ^{2}". Esta cantidad mide cuánto se acercan en promedio las predicciones del algoritmo de aprendizaje al objetivo (sobre todos los posibles conjuntos de entrenamiento del mismo tamaño).

"Varianza" Esta cantidad recoge cuánto fluctúan las predicciones del algoritmo de aprendizaje para los diferentes conjuntos de entrenamiento del tamaño dado.

Nótese que construyendo un clasificador agregado y reemplazando C con C_A se reduce la varianza a cero, pero no hay garantías de que se reduzca el sesgo. En realidad, se pueden dar ejemplos donde el sesgo aumentará, como el propuesto por Tibshirani (1996): Supóngase que y=1 para todo x, y el clasificador C predice y=1 (para cualquier x) con probabilidad 0'4 y predice y=0 (para cualquier x) con probabilidad 0'6. Entonces PE(Y, C) =0'6 y PE(Y, C_A)=1, por lo tanto, en este caso el error de predicción del clasificador agregado será superior al del clasificador individual.

Descomposición del error de un clasificador. Tibshirani (1996)	
Riesgo de Bayes	Es el error cometido por el clasificador óptimo de Bayes
Sesgo	Mide cuánto se separa el clasificador agregado del clasificador óptimo de Bayes.
Varianza	Mide cuánto se separa el clasificador individual del clasificador agregado.

Tabla 5.1. Descomposición del error.

5.2.1. ESTIMACIONES BOOTSTRAP DEL SESGO Y DE LA VARIANZA

Una vez establecida la descomposición del error de predicción de un clasificador en los tres términos mencionados, riesgo de Bayes, sesgo y varianza, hay que intentar

calcular estas cantidades, pero esta labor no es, a priori, sencilla. Sin embargo, utilizando el método bootstrap, se puede lograr una estimación basada en la muestra de la descomposición del error de predicción. La aproximación bootstrap genérica utiliza la distribución empírica $\hat{F}$ para sustituir a F. Pero, en este caso hay que estimar dos distribuciones, la distribución del conjunto de entrenamiento T y la distribución del conjunto de prueba F. Efron y Tibshirani (1995) rechazan el uso de $\hat{F}$ para ambas estimaciones, ya que conduce a una infravaloración del sesgo, puesto que las bases del conjunto de entrenamiento y de prueba se solapan. Para evitar esto, se puede utilizar el sistema bootstrap dejando uno fuera de Efron y Tibshirani (1995). En primer lugar, se recoge la estimación bootstrap del término correspondiente a la varianza.

$$Var(C)=E_T E_F[C(x,T) \neq C_A(x)] \tag{5.10}$$

Sea $\hat{F}_{(i)}$ la distribución obtenida asignando un peso 1/(n-1) a todos los ejemplos menos a x_i, al que se asigna peso cero. Entonces la estimación es

$$\begin{aligned} \hat{Var}(C) &= E_{\hat{F}} E_{\hat{F}_{(i)}}[C(x,T_i^b) \neq C_A(x,\hat{F}_{(i)})] = \\ &= \frac{1}{n}\sum E_{\hat{F}_{(i)}}[C(x_i,T_i^b) \neq C_A(x_i,\hat{F}_{(i)})] \end{aligned} \tag{5.11}$$

donde T_i^b es una réplica bootstrap del conjunto de entrenamiento obtenida a partir de $\hat{F}_{(i)}$, además se ha hecho explícita la dependencia de $C_A(x)$ en $\hat{F}_{(i)}$. Se puede estimar $\hat{Var}(C)$ utilizando un conjunto de muestras de Monte Carlo (MC), siguiendo los siguientes pasos:

1. Se obtienen conjuntos de entrenamiento bootstrap $T^1, T^2, ..., T^B$, muestreando con reemplamiento del conjunto de entrenamiento T. Se calcula el clasificador $C(x, T^b)$ en cada uno de ellos.

2. Sea V_i el conjunto de índices de los conjuntos de entrenamiento bootstrap que no contienen la observación *i*-ésima. Para cada *i*, se construye el clasificador agregado $C_A(x,\hat{F}_{(i)})$ a partir de los clasificadores $C(x, T^b)$ para b0V_i, obteniéndose:

$$\hat{C}_A(x,\hat{F}_{(i)}) = \sum_{b \in V_i} C(x_i,T^b)/B_i \tag{5.12}$$

donde B_i es el número de muestras bootstrap en V_i. La estimación de la Var(C) se obtiene

$$\hat{Var}_{Boot}(C) = \frac{1}{n}\sum_{i=1}^{n}\sum_{b\in V_i}[C(x_i, T^b) \neq C_A(x_i, \hat{F}_{(i)})] / B_i \quad \textbf{(5.13)}$$

Por otro lado, la estimación bootstrap del clasificador agregado $C_A(x) = E_T[C(x, T)]$ es

$$\hat{C}_A(x) = E_{\hat{F}}[C(x, T^b)] \quad \textbf{(5.14)}$$

Esta estimación bootstrap del clasificador agregado coincide con el clasificador bagging de Breiman (1996). La estimación bootstrap dejando uno fuera del error para $\hat{C}_A$ es

$$\hat{Err}^{(1)}(\hat{C}_A) = \frac{1}{n}\sum_{i=1}^{n} E_{\hat{F}_{(i)}}[y_i \neq C_A(x_i, \hat{F}_{(i)})] \quad \textbf{(5.15)}$$

Esta expresión se estima utilizando muestras de Monte Carlo (MC) de la misma manera que se realiza la estimación de la varianza, es decir

$$\hat{Err}^{(1)}_{Boot}(\hat{C}_A) = \frac{1}{n}\sum_{i=1}^{n}\sum_{b\in V_i}[y_i, C_A(x_i, \hat{F}_{(i)})] / B_i \quad \textbf{(5.16)}$$

Sin embargo, la estimación de la componente sesgo es más problemática. Utilizando la definición de sesgo Sesgo(C) / PE (C*, C_A) y denotando por PE(C*) el error de predicción del clasificador óptimo de Bayes.

$$\hat{Sesgo}(C) = \hat{Err}^{(1)}(\hat{C}_A) - PE(C^*) \leq \hat{Err}^{(1)}(\hat{C}_A) \quad \textbf{(5.17)}$$

El término $\hat{Err}^{(1)}(\hat{C}_A)$ proporciona un límite superior aproximado para el sesgo(C). Un límite más ajustado puede obtenerse estimando el riesgo de Bayes, PE(C*), pero esto es difícil de estimar y no está dentro de los objetivos de este trabajo. En este tipo de análisis, lo que interesa es ver la efectividad de distintas técnicas que intentan reducir el error de predicción de un clasificador, distinguiendo si afecta únicamente al sesgo o a la varianza o, por el contrario, consiguen reducir ambos términos. En estas circunstancias, si el término sesgo incluye o no el riesgo de Bayes no afectará a las conclusiones, ya que el riesgo de Bayes no puede ser reducido, por tanto, toda la reducción que se consiga en el sesgo estimado se deberá, exclusivamente, al verdadero sesgo y no al riesgo de Bayes. Por todo ello, las estimaciones del sesgo incluirán el riesgo de Bayes, como sucede también en Bauer y Kohavi (1998).

5.3. ESTUDIO DE LA INESTABILIDAD DE LOS CLASIFICADORES

Después de ver la descomposición sesgo varianza del error de generalización, en este apartado se estudia la inestabilidad de los clasificadores. En los problemas de clasificación es habitual encontrarse con conjuntos de datos de reducido tamaño, especialmente en el caso de problemas reales, es lo que se conoce como el problema de tamaño muestral pequeño. Esto sucede cuando el tamaño del conjunto de entrenamiento es pequeño en comparación con la dimensión de los datos, es decir, el número de características que describen cada ejemplo. En estos casos, puede ser difícil construir un buen clasificador, así por ejemplo en el caso del discriminante lineal será difícil construir una buena función discriminante, ya que las estimaciones de las medias de las clases y de la matriz de covarianzas pueden estar sesgadas. Construir el clasificador utilizando esas estimaciones puede conducir a una función discriminante que tenga un mal comportamiento. Además, cuando el número de ejemplos de entrenamiento es inferior a la dimensión de los datos, algunos clasificadores no pueden ser construidos, en general aquellos que utilizan la inversa de la matriz de covarianzas. Bajo estas condiciones, la estimación muestral de la matriz de covarianzas será una matriz singular, que no puede ser invertida. En cualquier caso, los clasificadores construidos en conjuntos de entrenamiento pequeños son normalmente sesgados y pueden tener una varianza grande. Por esto, se dice que estos clasificadores son inestables.

Para estudiar la inestabilidad de los clasificadores, es necesario establecer alguna forma de medir este concepto. De manera intuitiva la inestabilidad de un clasificador puede relacionarse a la desviación típica de su error de clasificación en el conjunto de prueba, si está disponible, o del conjunto de entrenamiento en el peor de los casos. De esta forma, se puede ver cuánto varía el error de clasificación cuando se consideran diferentes conjuntos de entrenamiento del mismo tamaño, o distintos valores iniciales de los parámetros utilizados en la construcción de un clasificador, por ejemplo, los pesos iniciales en una red neuronal. Sin embargo, la desviación típica del error de clasificación no tiene en cuenta directamente el tamaño del conjunto de entrenamiento o ciertos cambios en la composición de este conjunto. En realidad, la desviación típica

del error de clasificación no muestra la inestabilidad de un clasificador. Únicamente muestra la inestabilidad de su comportamiento (el error de clasificación) en el conjunto de datos disponible (conjunto de entrenamiento o de prueba). El error de clasificación no refleja necesariamente la inestabilidad del clasificador, por ejemplo, en el caso del análisis discriminante, el error de clasificación no muestra la inestabilidad de la función discriminante.

Cuando tanto el tamaño del conjunto de entrenamiento como la dimensión de los datos son pequeños, se pueden obtener clasificadores muy distintos en conjuntos de entrenamiento del mismo tamaño, aunque el comportamiento de estos clasificadores puede ser el mismo. Así mismo, cuando tanto el tamaño del conjunto de entrenamiento como la dimensión de los datos son grandes, el error de clasificación condicionado puede ser el mismo, sin embargo, los clasificadores serán distintos. Por lo tanto, para medir la inestabilidad de un clasificador y ver cómo éste varía, se debe utilizar otro procedimiento que tenga en cuenta el tamaño del conjunto de entrenamiento y su composición. Esto es especialmente importante cuando es necesario medir la inestabilidad de clasificadores construidos, por ejemplo, en réplicas bootstrap del conjunto de entrenamiento.

En Skurichina (2001) se propone una posible medida de la inestabilidad de un clasificador. Con esta *medida de inestabilidad*, se pretende tener en cuenta la influencia de los cambios en la composición del conjunto de entrenamiento, sobre la inestabilidad del clasificador en cuestión. Para realizar los cambios en el conjunto de entrenamiento se utiliza el ya mencionado método de muestreo bootstrap. Esta técnica parte de un conjunto de n elementos, y consiste en seleccionar n elementos extraídos aleatoriamente con reposición, de esta forma alguno de los elementos en el conjunto original aparecerán varias veces en la réplica bootstrap mientras que otros, por el contrario, no aparecen ninguna vez. Por tanto, si se parte del conjunto de entrenamiento $T= \{(x_1, y_1), ..., (x_n, y_n)\}$, se realizan réplicas bootstrap del tipo $T^b= \{(x_1^b, y_1^b), ..., (x_n^b, y_n^b)\}$. Es importante resaltar que en T^b algunos ejemplos estarán presentes una, dos, o incluso más veces, y otros pueden no aparecer ninguna vez en esa réplica bootstrap.

La medida de inestabilidad que propone Skurichina (2001), se realiza de la siguiente manera. Sea T= $\{(x_1, y_1), ..., (x_n, y_n)\}$ el conjunto de entrenamiento y $\tilde{T} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_m)$ el conjunto de prueba.

1. Construir el clasificador C en el conjunto de entrenamiento T y clasificar el conjunto de prueba $\tilde{T} \rightarrow C(\tilde{x}_i)$, donde $C(\tilde{x}_i) = y_i \quad (i = 1, ...,m)$, si $\tilde{x}_i$ es clasificado por C en la clase y_i.

2. Repetir para b=1, 2, ...,B

a) Tomar la muestra bootstrap T^b y construir el clasificador básico C_b en esta muestra bootstrap y clasificar el conjunto de prueba $\tilde{X} \rightarrow C_b(\tilde{x}_i)$, donde $C_b(\tilde{x}_i) = y_i \quad (i = 1, ...,m)$, si $\tilde{x}_i$ es clasificado por C_b en la clase y_i

b) Calcular la probabilidad de que la predicción sea distinta cuando se clasifica el conjunto de prueba utilizando C y C_b, $Prob\,(C_b(\tilde{x}_i) \neq C(\tilde{x}_i))$

3. La inestabilidad se calcula como

$$Inestabilidad = \frac{1}{B}\sum_{b=1}^{B} Prob\,(C_b(\tilde{x}_i) \neq C(\tilde{x}_i)) \qquad \textbf{(5.18)}$$

Skurichina (2001) muestra las ventajas de esta medida de inestabilidad en comparación con la desviación típica del error de clasificación, utilizando conjuntos de datos generados artificialmente. Medir la inestabilidad de un clasificador a través de la desviación típica del error de clasificación medio, tanto si se obtiene en el conjunto de prueba, error de generalización, como si se obtiene en el mismo conjunto de entrenamiento, error aparente, no proporciona resultados satisfactorios en el caso en el

que el conjunto de entrenamiento es de tamaño pequeño. La desviación típica del error de generalización medio es engañosa cuando los clasificadores, obtenidos en conjuntos de entrenamiento diferentes de tamaño pequeño, tienen un comportamiento igualmente malo en el conjunto de prueba. Sin embargo, estos clasificadores pueden en realidad ser muy distintos entre sí, por ejemplo, en el caso de los árboles de clasificación, árboles distintos pueden tener un comportamiento similar. En este caso, la desviación típica del error de generalización medio será relativamente pequeña, aunque el clasificador es muy inestable.

De la misma manera, la desviación típica del error de generalización medio puede llevar a confusión cuando se tienen clasificadores distintos que se comportan muy bien, en el caso más extremo todos ellos con error aparente cero. A pesar de que el clasificador sea inestable, la desviación típica del error aparente será igual a cero, lo que indicaría una alta estabilidad del clasificador. La medida de inestabilidad de la ecuación (5.18) no presenta estos inconvenientes siendo más objetiva para medir la inestabilidad del clasificador.

Se puede ver que el comportamiento de los clasificadores y su estabilidad están correlacionados. Por regla general, los clasificadores más inestables tienen un peor comportamiento y los clasificadores más estables obtienen mejores resultados. Por tanto, es lógico pensar que para mejorar el comportamiento de un clasificador inestable, éste debe ser estabilizado.

Según Breiman (1996), los árboles de clasificación y las redes neuronales son métodos inestables, mientras que los métodos del vecino más próximo y el discriminante lineal son estables. Los métodos como los árboles tienen una varianza alta, pero aciertan en promedio, es decir, son bastante insesgados, la clase correcta es habitualmente la ganadora si se aplica el voto mayoritario para la agregación de varios de ellos. Los métodos estables, como el discriminante lineal, consiguen su estabilidad teniendo un conjunto de modelos muy limitado que ajustar a los datos. El resultado es

una varianza baja, pero si no se puede representar correctamente los datos con el conjunto de modelos disponibles, puede obtenerse un sesgo elevado.

Para calcular la medida de inestabilidad de un clasificador descrita en la ecuación (5.18), Skurichina (2001) utiliza 25 réplicas bootstrap del conjunto de entrenamiento y construye un clasificador básico en cada uno de ellos. Este proceso se repite 50 veces en conjuntos de entrenamiento independientes del mismo tamaño para obtener el promedio de los resultados.

Teóricamente la medida de inestabilidad debe utilizar el conjunto de prueba, ya que de esta forma el resultado es más objetivo que la inestabilidad medida en el conjunto de entrenamiento. La inestabilidad muestra cuánto cambia el clasificador con los cambios en la composición del conjunto de entrenamiento. Cuando se considera la inestabilidad medida en el conjunto de prueba se puede ver que, para cualquier conjunto de datos, los clasificadores alcanzan la máxima inestabilidad para tamaños pequeños del conjunto de entrenamiento. Si se dispone de pocas observaciones para realizar el aprendizaje, incluso pequeños cambios en la composición del conjunto de entrenamiento pueden afectar mucho al clasificador resultante, por ejemplo, al árbol de clasificación o a la función discriminante. Si se incrementa el número de ejemplos de entrenamiento por encima del tamaño crítico, la dimensión de los datos, los clasificadores se hacen más estables y su error de generalización disminuye. La composición del conjunto de entrenamiento resulta menos importante cuando el tamaño del mismo es suficientemente grande como para representar correctamente la verdadera distribución del problema.

La inestabilidad medida en el conjunto de entrenamiento está más sesgada, porque el mismo conjunto de entrenamiento es utilizado para construir el clasificador y también para estimar su inestabilidad. Sin embargo, en la práctica el conjunto de prueba no puede ser utilizado y habrá que utilizar alguna de las técnicas expuestas en el capítulo 2 para la estimación del error. Además, se puede observar que la inestabilidad medida en el conjunto de entrenamiento presenta el mismo comportamiento que si se mide en el conjunto de prueba, la inestabilidad de los clasificadores disminuye a medida que

aumenta el tamaño del conjunto de entrenamiento. Es decir, la inestabilidad medida en el conjunto de entrenamiento y de prueba fluctúan casi en paralelo.

En general, la inestabilidad de un clasificador dependerá de varios factores como la complejidad del mismo, la distribución de los datos utilizados para construir el clasificador y la sensibilidad de éste al tamaño y a la composición del conjunto de entrenamiento.

5.4. ¿POR QUÉ UTILIZAR COMBINACIONES DE CLASIFICADORES?

En este apartado se recogen algunas justificaciones de la superioridad de los clasificadores combinados sobre los clasificadores individuales. La idea de partida es utilizar un conjunto de clasificadores para obtener una mayor precisión de la que cada uno de éstos logra de manera individual. Cada método de clasificación se basa en conceptos o procedimientos de estimación diferentes, además puesto que todos los métodos de clasificación tienen algún punto fuerte o ventaja sobre la regla por defecto[21], es lógico intentar aunar las mejores propiedades de cada uno de ellos combinándolos de alguna manera. Para ello, en ocasiones se trabaja con la clase predicha por los clasificadores individuales mientras que en otras, lo que se utiliza son las probabilidades condicionadas asignadas a cada clase por los distintos clasificadores.

Combinar la salida de varios clasificadores es útil únicamente si hay desacuerdo entre ellos. Obviamente combinando varios clasificadores idénticos no se obtiene ningún beneficio. Hansen y Salomon (1990) probaron que, si la tasa de error media para una observación es menor del cincuenta por ciento y los clasificadores utilizados en el comité son independientes en la producción de sus errores, el error esperado para una observación puede reducirse a cero cuando el número de clasificadores combinado se acerca a infinito.

---

[21] La regla por defecto consiste en asignar a todas las observaciones la clase mayoritaria en el conjunto de entrenamiento, sin tener en cuenta ningún atributo o característica de las que presenta esa observación.

Por su parte, Krogh y Vedelsby (1995) probaron más tarde que el error conjunto puede dividirse en un término que mide el error de generalización medio de cada clasificador individual, y otro que recoge el desacuerdo entre los clasificadores. Lo que ellos demostraron formalmente fue que la combinación ideal consiste en clasificadores con alta precisión, que estén el mayor número de veces posible en desacuerdo.

Volviendo al planteamiento de Hansen y Salomon, incluso el método más sencillo de combinación, el voto mayoritario, y bajo el supuesto de que los clasificadores son independientes entre sí, se puede comprobar, para un problema dicotómico, como la precisión del conjunto es superior a la de los clasificadores individuales, siempre que estos cometan un error inferior al cincuenta por ciento. En concreto, puede verse como la probabilidad de que fallen más de la mitad de todos los clasificadores individuales considerados. La figura 5.1 muestra cómo evoluciona el error del conjunto, en función del número de clasificadores utilizados en la combinación, para diferentes valores de la probabilidad de error de los clasificadores individuales ($\mathbf{g}$). Se considera únicamente las combinaciones con un número impar de clasificadores, como suele hacerse en la práctica para evitar empates. A medida que aumenta el número de clasificadores utilizados, aumenta la precisión del conjunto para un valor dado de la precisión individual.

Por su parte, la figura 5.2 recoge cómo varía el error del conjunto, en función de la precisión de los clasificadores base, entre 0 y 0,5, para distintos tamaños de la combinación. Como se podía esperar, cuanto menor es la precisión de los clasificadores básicos, menor es también la del conjunto para un mismo tamaño del conjunto. Cualquiera que sea el número de clasificadores utilizados, cuando su precisión alcanza el valor 0,5 no se obtiene ningún beneficio a partir de esta combinación.

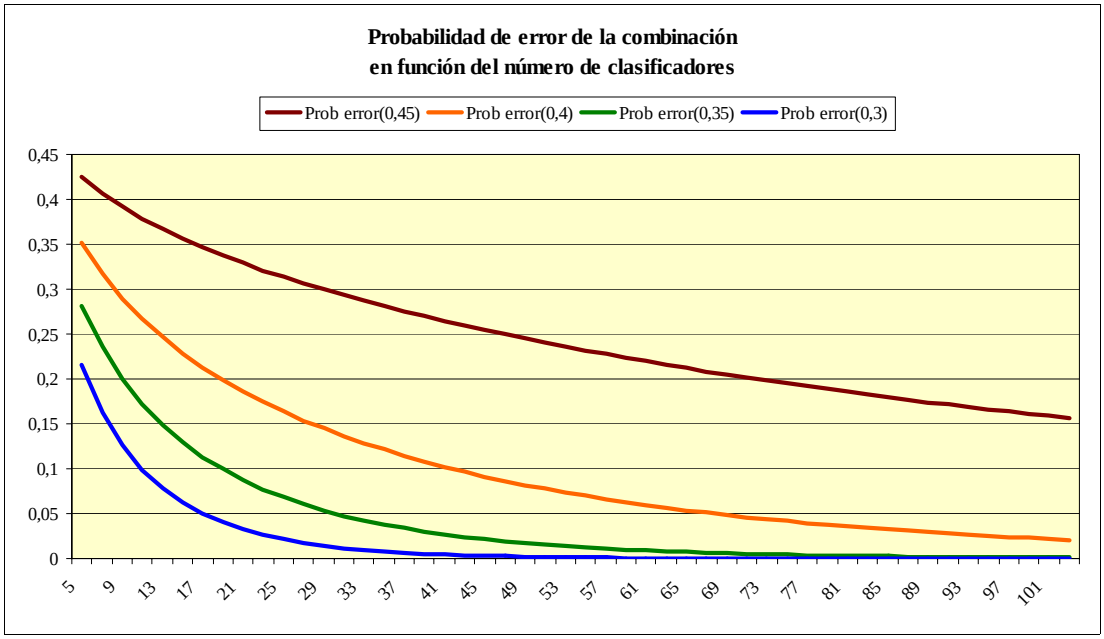


Figura 5.1. Probabilidad de error de la combinación mediante voto mayoritario en función del número de clasificadores utilizados en la misma.

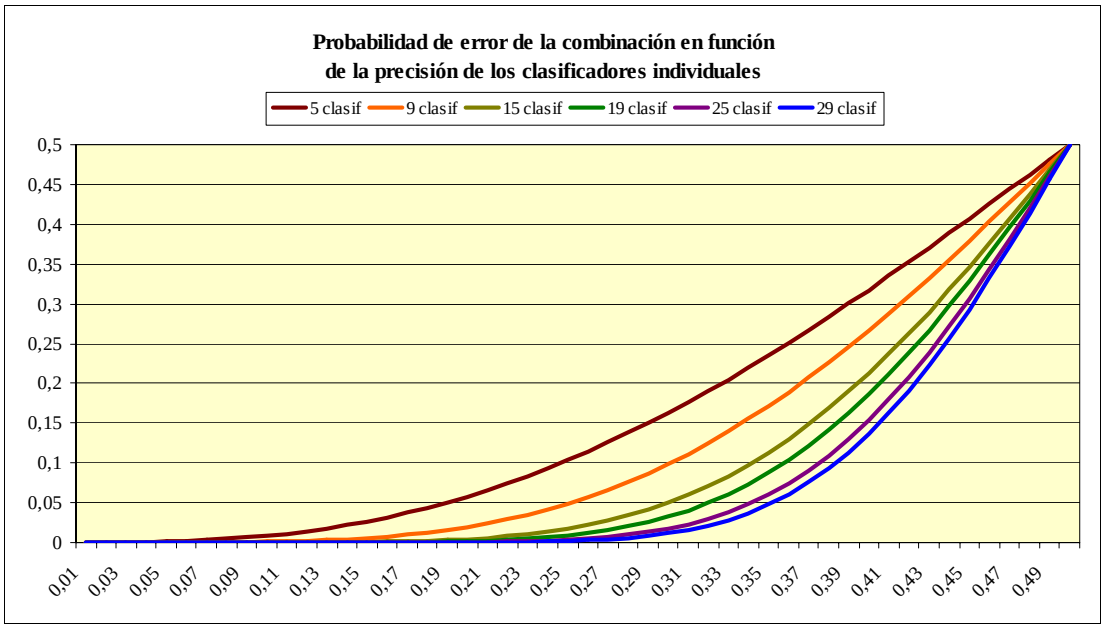


Figura 5.2. Probabilidad de error de la combinación mediante voto mayoritario en función de la precisión de los clasificadores individuales.

A pesar de la sencillez y claridad de esta explicación, está basada en supuestos muy restrictivos en la práctica, como es la independencia entre los clasificadores básicos. Además, el planteamiento para el cálculo de la probabilidad de error del conjunto, sólo es válido para este método de combinación. Por todo ello, deben existir otras razones

más consistentes que expliquen la superioridad del conjunto sobre los clasificadores individuales. En Dietterich (2000) se plantean las tres razones siguientes.

La primera razón Dietterich la denomina razón estadística. Un sistema de aprendizaje puede entenderse como la búsqueda, dentro de un determinado espacio de hipótesis, de aquella que sea la más adecuada. En la práctica, es habitual encontrarse con conjuntos de datos cuyo tamaño es demasiado pequeño en comparación con el tamaño del espacio de hipótesis. En estos casos, el método de clasificación puede encontrarse con varias hipótesis distintas, que presentan una precisión similar en el conjunto de datos disponible. Si se combinan las distintas hipótesis, se pueden promediar los resultados y reducir el riesgo de elegir un clasificador que obtenga una menor precisión a la hora de generalizar.

En segundo lugar considera una razón de computación. Viene causada por la propia naturaleza de determinados sistemas de clasificación que realizan algún tipo de búsqueda local, por lo que pueden quedar atrapados en un óptimo local. En este tipo de clasificadores se encuentran entre otros las redes neuronales, que utilizan el método del gradiente descendente para minimizar una función de coste sobre las observaciones de entrenamiento, y los árboles de clasificación, que utilizan una regla de corte secuencial en el desarrollo del árbol. Incluso en casos donde el tamaño del conjunto de entrenamiento es suficiente como para que no exista el problema anterior, sigue siendo computacionalmente difícil para el sistema de aprendizaje encontrar la mejor hipótesis. En conclusión, la combinación de clasificadores, obtenidos realizando la búsqueda local desde puntos iniciales distintos, logrará una mejor aproximación a la hipótesis o función objetivo buscada que cualquiera de los clasificadores individuales.

La tercera es la razón de representación. En muchos problemas de clasificación, la verdadera función no se puede representar mediante ninguna de las hipótesis existentes en el espacio de hipótesis disponibles. Sin embargo, mediante combinaciones de hipótesis relativamente sencillas de ese espacio se puede llegar a una mejor aproximación a la función real, ya que se logra ampliar el espacio de funciones

representables. Por ejemplo, utilizando una suma ponderada de las hipótesis simples. Esta cuestión es aplicable incluso a aquellos métodos de clasificación que teóricamente tienen capacidad para representar cualquier función. Por ejemplo, las redes neuronales o los árboles de clasificación, que son métodos muy flexibles y con una gran capacidad de representación cuando el número de observaciones es ilimitado, pero en la práctica lo habitual es enfrentarse a conjuntos de datos limitados, por lo que incluso estos sistemas trabajan con un conjunto de hipótesis también limitado, y detendrán la búsqueda cuando encuentren una hipótesis que se ajuste correctamente a los datos disponibles.

Estas son las tres razones fundamentales que señala Dietterich para justificar la superioridad de la combinación de clasificadores sobre los individuales. Ahora bien, no se debe olvidar que también presentan algunas desventajas como son la pérdida de comprensibilidad (estructura más compleja y mayor tamaño), la mayor lentitud en la construcción de la combinación (se necesita una mayor capacidad de memoria), y también se tarda más a la hora de clasificar una nueva observación.

5.5. LA PERSPECTIVA BAYESIANA SOBRE LA AGREGACIÓN DE CLASIFICADORES

Para finalizar el capítulo, se recoge la visión bayesiana acerca de la agregación de métodos de clasificación, o de manera más general sobre la agregación de modelos. Desde una perspectiva bayesiana, la agregación de modelos es la opción más natural ya que, de esta forma, no sólo se tiene en cuenta el riesgo o incertidumbre inherente al conjunto de datos con el que se trabaja, sino que, también se considera el riesgo asociado a la elección de un único modelo de entre las distintas opciones existentes.

En primer lugar, hay que distinguir si la finalidad es la selección explícita de un modelo, ver cuál es el que presenta unas características más adecuadas para una determinada tarea, o por el contrario, el objetivo es la predicción en futuras observaciones a partir de lo aprendido durante el entrenamiento en el conjunto de datos disponible. Así, cuando la finalidad no es la selección de un modelo en concreto sino

la obtención del máximo rendimiento a partir del conjunto de datos disponible, la opción más adecuada desde un punto de vista bayesiano es la agregación o combinación de modelos.

Esto en terminología bayesiana se conoce como promediación bayesiana de modelos[22], que tiene en cuenta la incertidumbre debida a los modelos. En concreto, esta técnica actúa de la siguiente manera. Si *y* es la clase a predecir, su distribución a posteriori dado el conjunto de datos T será:

$$Prob\,(y \mid T) = \sum_{b=1}^{B} Prob(y \mid M_b\,, T)\; Prob(M_b \mid T) \qquad \textbf{(5.19)}$$

de esta forma, se promedian las distribuciones a posteriori bajo cada uno de los modelos considerados, ponderada por la probabilidad a posteriori de cada modelo. En esta ecuación M_1 , M_2 ,....., M_B son los modelos considerados.

Este marco general de promediación bayesiana de modelos, se puede aplicar igualmente al caso que interesa en este trabajo de los métodos de clasificación. Donde los modelos M_1, M_2,....., M_B , son los procedimientos de clasificación, que serán combinados en aras a la reducción del riesgo asociado a la elección de un método individual.

[22] En inglés, Bayesian Model Averaging (BMA). Para una descripción detallada de esta técnica véase Hoeting, Madigan y Raftery (1999): “Bayesian Model Averaging. A Tutorial”. Statistical Science, 14:4.

CAPÍTULO 6

MÉTODOS DE COMBINACIÓN DE CLASIFICADORES

6.1. INTRODUCCIÓN

En el capítulo anterior, se han estudiado algunos aspectos relacionados con el comportamiento y las propiedades de los clasificadores individuales o básicos, como son el sesgo, la varianza y la inestabilidad. Además, se ha puesto de manifiesto la superioridad de los clasificadores combinados sobre los individuales.

Por un lado, se ha descompuesto el error de generalización en la suma de tres términos no negativos como son el riesgo de Bayes, el sesgo y la varianza. El primero de ellos, recoge el error inevitable o inherente en un conjunto de datos. El sesgo mide el error persistente del método de clasificación, es decir, el error que se mantendría incluso si se tuviese un conjunto infinito de clasificadores entrenados de manera independiente. El término correspondiente a la varianza mide el error debido a las fluctuaciones que se producen al generar un clasificador individual.

Por otro lado, se ha estudiado la inestabilidad de los clasificadores, que hace referencia a los cambios que se producen en el clasificador ante pequeños cambios en el conjunto de entrenamiento, si el clasificador sufre grandes cambios, se dice que es inestable. En general, se consideran estables el análisis discriminante lineal y el método del vecino más próximo, mientras que los árboles de clasificación y las redes neuronales son considerados como métodos inestables. La agregación de modelos consigue clasificadores más estables.

Entre las razones que explican la superioridad de los clasificadores combinados sobre los individuales, destacan las tres razones propuestas por Dietterich (2000). La razón estadística se basa en la eliminación del riesgo asociado a la elección de un clasificador frente al resto. La razón de computación plantea la posibilidad de que los sistemas de clasificación que realizan una búsqueda local, puedan quedar atrapados en un óptimo local y verse imposibilitados para alcanzar el óptimo global. Finalmente, la razón de representación aduce la imposibilidad de representar la verdadera función con ninguna de las hipótesis disponibles.

Por último, se recoge la perspectiva bayesiana, que también defiende la agregación de modelos cuando el objetivo no es la selección explícita de un modelo, sino extraer el mejor rendimiento posible al conjunto de datos disponible para la predicción que se desea realizar.

En este capítulo, se aborda el estudio de los clasificadores combinados centrándose fundamentalmente en el método boosting, pero recogiendo además una clasificación de los métodos de combinación de clasificadores e introduciendo también el método bagging y los bosques aleatorios. A continuación se plantea la estructura del capítulo.

En primer lugar, se recogen algunas de las clasificaciones de los métodos de combinación de clasificadores. Debido a la importancia que recientemente ha cobrado la agregación de métodos en el reconocimiento de patrones, en la última década se han propuesto varias clasificaciones de los métodos de combinación de los sistemas de

aprendizaje. Se citan algunas clasificaciones de estos métodos realizadas a partir de ciertas características que presentan los mismos, como son si los clasificadores combinados utilizan la misma representación o no para las observaciones de entrada, la arquitectura de la agregación, si selecciona los clasificadores básicos o simplemente los combina, y si el método genera nuevos clasificadores individuales o se limita a combinar los que ya existen.

A continuación, se analiza el sistema bagging, que se basa en la agregación de un conjunto de clasificadores básicos, cada uno de ellos es entrenado sobre una réplica bootstrap del conjunto de entrenamiento original y la decisión final de clasificación se basa en el voto de cada uno de estos clasificadores básicos. Este sistema aprovecha las ventajas de la técnica de remuestreo bootstrap, así como las derivadas de la agregación, de ahí su nombre, Bagging, del inglés “bootstraping and aggregating”.

En tercer lugar, se expone el método boosting, cuyo objetivo es mejorar la precisión de un sistema de clasificación dado. Para ello, primero se entrena un clasificador cuya precisión en el conjunto de entrenamiento supere al menos la de la regla por defecto (0,5 en el caso dicotómico con probabilidades a priori iguales). Posteriormente, se añaden nuevos clasificadores básicos para formar un combinado cuya precisión en el conjunto de entrenamiento sea arbitrariamente alta. En ese caso, se dice que el comportamiento del clasificador ha sido potenciado[23]. De manera esquemática, puede considerarse que boosting entrena sucesivamente a los clasificadores básicos con versiones modificadas del conjunto de entrenamiento, que son más informativas dado el conjunto existente de clasificadores básicos. La clasificación final de una observación se basa en las salidas de los clasificadores básicos.

Por último, para acabar el capítulo, se presenta el método del bosque aleatorio, donde se genera un grupo de árboles para su posterior combinación. Para intentar que no estén muy relacionados entre sí, se introduce aleatoriedad en la generación de estos árboles,

[23] En inglés, para decir que el clasificador ha sido potenciado, se emplea la palabra boosted, de ahí el nombre del método.

de tal forma que cada árbol va a ser una función del conjunto de entrenamiento, pero también de un vector aleatorio, que influirá en su desarrollo.

6.2. CLASIFICACIÓN DE LOS MÉTODOS DE COMBINACIÓN

Debido a la importancia que recientemente se le ha dado a la agregación de métodos en el reconocimiento de patrones, en la última década se han propuesto varias clasificaciones de los métodos de combinación de los sistemas de aprendizaje. Aunque existen otras, se han seleccionado sólo las que se consideran más importantes.

En primer lugar, se puede citar la clasificación propuesta por Kittler (1998) que distingue según el punto de vista del análisis, básicamente dos escenarios de combinación. En el primero, todos los clasificadores utilizan la misma representación en los patrones de entrada u observaciones. Por ejemplo, un conjunto de clasificadores del vecino más próximo que utilizan el mismo vector de variables, pero distinto valor de *k* o en la distancia empleada para determinar el número de vecinos a considerar. O un conjunto de redes neuronales con la misma estructura, pero con distinto vector de pesos obtenidos con diferentes estrategias de aprendizaje. De esta forma, cada clasificador, para una observación determinada, se puede entender que obtiene una estimación de la misma probabilidad de clase a posteriori.

En el segundo caso, cada clasificador utiliza su propia representación de las observaciones de entrada. Es decir, las medidas extraídas de las observaciones son únicas para cada clasificador. Una importante aplicación de la agregación de clasificadores en este escenario es la integración de distintos tipos de características o atributos de las observaciones. En esta situación, ya no es posible considerar las probabilidades a posteriori calculadas, como estimación de un mismo valor funcional, puesto que los sistemas de clasificación trabajan en diferentes espacios de medida.

En Dietterich (2000) se establecen los siguientes grupos de los métodos de combinación diferenciando entre aquellos que realizan una votación de tipo bayesiano,

los que modifican los ejemplos de entrenamiento, los que modifican las variables, los que modifican las clases posibles, y por último, los que aleatorizan el sistema de aprendizaje.

En tercer lugar, Lam (2000) propone agrupar estos métodos según la arquitectura de la agregación, distinguiendo entre si ésta se realiza en serie, en paralelo o de forma jerárquica. En Jain, Duin y Mao (2000) se recoge la separación en función de si los clasificadores básicos son seleccionados o no por el algoritmo de combinación, diferenciando entre los métodos de combinación orientados a la selección y aquellos orientados a la combinación.

Por último, Masulli y Valentini (2002) utilizan una clasificación parecida a la anterior, en función de si el algoritmo de combinación actúa o no sobre los clasificadores básicos modificándolos, de esta forma distingue entre métodos generadores y no generadores de combinación. Los *métodos generadores* crean conjuntos de clasificadores básicos actuando sobre el propio sistema de clasificación, o sobre el conjunto de datos de entrenamiento, intentando activamente mejorar la diversidad y la precisión de los clasificadores básicos. Los *métodos no generadores* se limitan a combinar un conjunto dado de clasificadores básicos posiblemente bien diseñados, es decir, no generan nuevos clasificadores básicos, sino que intentan combinar de la mejor forma posible los ya existentes. A continuación se hace una referencia más detallada a esta clasificación.

6.2.1. Métodos no generadores

En **los métodos no generadores** los clasificadores básicos se unen mediante un procedimiento de combinación, que depende de su capacidad de adaptación a las observaciones de entrada, y de las necesidades de la salida que proporcionan los sistemas de aprendizaje individuales. Es decir, el tipo de combinación depende del tipo de salida. Si sólo se dispone de la clase asignada, o si las salidas continuas son difíciles de manejar, entonces se utiliza el voto mayoritario, ésta es la forma más sencilla de

combinar varios clasificadores y consiste en asignar a una observación la clase que predicen la mayoría de los clasificadores. Es decir, cada nueva observación hay que presentarla ante cada clasificador individual para ver qué clase le asigna. Una vez que todos los clasificadores individuales han dado su predicción, se asigna a esa nueva observación la clase que mayoritariamente se haya repetido. En caso de empate se asigna la clase con mayor probabilidad a priori, y si ésta no se conoce, la clase mayoritaria en el conjunto de entrenamiento o, en el peor de los casos, se resuelve al azar.

El problema reside en que de esta forma, todos los clasificadores del comité tienen la misma importancia y no se tiene en cuenta la mayor o menor precisión del mismo a la hora de la generalización. Además, el número de clasificadores incluido será un factor crítico, ya que demasiados clasificadores poco precisos pueden llevar a equivocar la decisión final y, por tanto, a disminuir la precisión del conjunto.

Este procedimiento puede mejorarse asignando un peso a cada clasificador individual, de tal forma que se optimiza el comportamiento del clasificador combinado en el conjunto de entrenamiento. Si los clasificadores básicos proporcionan las probabilidades a posteriori de las clases, se pueden agregar utilizando operadores sencillos como el mínimo, el máximo, la media, la mediana, el producto o la media ponderada.

6.2.2. Métodos generadores

Los métodos generadores intentan mejorar la precisión global de la combinación mediante la actuación directa sobre la precisión y la diversidad de los clasificadores base. Dentro de este grupo, se pueden distinguir subgrupos, en función de las estrategias que utilizan para conseguir mejorar los clasificadores básicos.

1. Métodos de remuestreo. Las técnicas de remuestreo vistas en el apartado 2.6 de este trabajo, pueden utilizarse para extraer muestras distintas a partir del conjunto de

datos original, de tal forma que los clasificadores individuales entrenados en cada una de ellas sean utilizados posteriormente en la combinación. De entre las técnicas de remuestreo, la que más se utiliza para este propósito es la técnica *bootstrap*, para obtener muestras con reemplazamiento del mismo tamaño que el conjunto de datos original, de tal forma que se obtienen distintos conjuntos de entrenamiento. Estas técnicas son especialmente aconsejables para sistemas de clasificación inestables, es decir, métodos muy sensibles a pequeños cambios en el conjunto de entrenamiento, como pueden ser los árboles de clasificación y las redes neuronales. Dentro de estos métodos de combinación que utilizan el remuestreo los más utilizados son los métodos *Bagging* y *Boosting*, de los que se hablará más adelante.

2. Métodos de selección de variables. Estos métodos reducen el número de características utilizadas en los clasificadores básicos, de esta forma se hace frente al problema de la maldición de la dimensión (Skurichina, 2001, p.156), que consiste en la escasez de ejemplos en relación al número de variables que describen cada una de las observaciones. Estos métodos de subespacios de características actúan dividiendo el conjunto de atributos, utilizando cada subconjunto para entrenar un clasificador básico. El método más utilizado es el Método del Subespacio Aleatorio[24] propuesto por Ho (1998), que selecciona aleatoriamente un subconjunto de características donde se entrena el clasificador base. De esta forma, se obtiene un subespacio aleatorio del espacio original de características y se construye el clasificador en ese subespacio. La agregación se realiza normalmente mediante voto ponderado por la precisión de los clasificadores individuales. En Skurichina (2001), se muestra que este método es efectivo para clasificadores cuya curva de aprendizaje[25] es decreciente, y que estén construidos en conjuntos de entrenamiento de tamaño pequeño y crítico.

3. Métodos de prueba y selección. Se basan en la idea de seleccionar los clasificadores básicos durante el proceso de creación de la combinación. Aunque existen

[24] Este método se conoce en inglés como Random Subspace Method (RSM).

[25] La curva de aprendizaje representa conjuntamente el error de generalización y el tamaño del conjunto de entrenamiento.

otras alternativas, cabe destacar la selección hacia delante y hacia atrás, a imitación de las estrategias que se siguen para la selección de variables en algunos sistemas como el análisis discriminante. Consiste en un proceso secuencial, donde en cada paso sólo se incluye (o se extrae) un nuevo clasificador del comité, si esto conlleva una reducción en el error.

4. Métodos aleatorios de agregación. Estos procedimientos aleatorizan el algoritmo de aprendizaje para generar combinaciones de sistemas de aprendizaje. Un ejemplo sería iniciar con valores aleatorios los pesos en el algoritmo de retropropagación, obteniendo clasificadores distintos para utilizarlos en la combinación. Algunos resultados experimentales muestran que aleatorizar los sistemas de aprendizaje utilizados para generar los clasificadores básicos de la combinación, mejora el comportamiento de los clasificadores individuales no aleatorios, Dietterich (2000). Entre estos métodos se encuentran los bosques aleatorios, que se verán en el apartado 6.5.

6.3. BAGGING

Uno de los problemas más habituales a la hora de establecer un clasificador para un conjunto de datos, es el tamaño limitado del conjunto de ejemplos de entrenamiento. Aunque este problema afecte especialmente a los métodos paramétricos, supone un reto para cualquier clasificador. Cuanto más pequeño sea el conjunto de datos disponibles, menos seguro se puede estar de que este conjunto represente fielmente a la población total. En general, los clasificadores construidos en conjuntos pequeños pueden estar sesgados y presentarán una elevada varianza en la probabilidad de clasificación errónea. Se dice en este caso que el clasificador es inestable.

En muchos casos no se puede disponer de más observaciones y, por tanto, el conjunto de entrenamiento está limitado. Existen diversas técnicas que intentan obtener clasificadores más estables, y actualmente éste es uno de los campos de investigación abiertos en el ámbito de los sistemas de clasificación.

Una posible solución es utilizar el sistema Bagging (**B**ootstrapping and **agg**regat**ing**). De igual forma que la estimación bootstrap de los parámetros de la distribución de los datos es más precisa y robusta que la estimación tradicional, se puede pensar en el uso de esta técnica para conseguir, una vez combinado, un clasificador con mejores propiedades.

Bagging fue propuesto por Breiman en 1996 y se basa en los métodos de bootstrapping y de agregación. Tanto los métodos de bootstrapping, como los de agregación, presentan propiedades beneficiosas. Bootstrapping consiste en obtener muestras aleatorias con reemplazamiento de igual tamaño que el conjunto original. Partiendo del conjunto de entrenamiento T= $\{(x_1, y_1), ..., (x_n, y_n)\}$ donde Y toma los valores{-1, 1}, mediante la extracción aleatoria con reemplazamiento con el mismo número de elementos que el conjunto original de *n* elementos, se obtienen B muestras bootstrap $T^b = \{(x_1^b, y_1^b), ..., (x_n^b, y_n^b)\}$ donde b=1, 2,.., B. En algunas de estas muestras se habrá eliminado o al menos reducido la presencia de observaciones ruidosas, por lo que el clasificador construido en ese conjunto presentará un mejor comportamiento que el clasificador construido en el conjunto original. Así pues, Bagging puede ser útil para construir un mejor clasificador cuando el conjunto de entrenamiento presente observaciones ruidosas.

El clasificador *combinado* obtiene frecuentemente mejores resultados que los clasificadores individuales utilizados para construir el clasificador final. Esto puede entenderse si se considera que al combinar los clasificadores individuales, se están combinando las ventajas de cada uno de ellos en el clasificador final.

En concreto el método Bagging se aplica del siguiente modo, se realizan B réplicas bootstrap, T^b, del conjunto de entrenamiento original T. Se construye un clasificador básico $C_b(x)$ en cada réplica T^b, con frontera de decisión igual a $C_b(x)$=0 dado que se considera el caso dicotómico, donde Y toma únicamente los valores {-1,1}. Por tanto si el valor de $C_b(x_i)$ es positivo, se le asigna a x_i el valor y=1, mientras que si $C_b(x_i)$ toma un valor negativo, se asigna a x_i el valor y= -1.

Una vez construidos los B clasificadores básicos se combinan usando el voto mayoritario, la clase predicha más frecuente, en la regla de decisión final.

$$C(x)=arg\max_{y\in\{-1,1\}}\sum_{b=1}^{B}\delta(C_b(x),y)\quad donde\ \delta(i,j)=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases} \tag{6.1}$$

La función $\delta(i, j)$, es conocida como la función delta de Kronecker, que toma el valor 1 si los términos *i* y *j* son iguales, y el valor cero si no lo son. El algoritmo 6.1 refleja esquemáticamente el método bagging.

Algoritmo 6.1. Bagging.

1. Repetir para b=1, 2, ...,B

   a) Realizar una réplica bootstrap T^b del conjunto de entrenamiento T.

   b) Construir un clasificador sencillo $C_b(x)$ en T^b.

2. Combinar los clasificadores básicos $C_b(x)$, b=1,2,...B, usando el voto

$$C(x)=arg\max_{y\in\{-1,1\}}\sum_{b=1}^{B}\delta(C_b(x),y)\quad donde\ \delta(i,j)=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases}$$

Existen otras funciones de combinación basadas en probabilidades a posteriori: mínimo, máximo, media, mediana y producto. Otra posibilidad, es promediar los coeficientes de los clasificadores básicos, para lo que se utiliza la función promedio, que presenta las siguientes ventajas, en primer lugar, no es necesario retener todos los resultados de la clasificación, sólo los coeficientes. En segundo lugar, obtiene un único clasificador con el mismo número de parámetros que los clasificadores básicos.

Cuando se extrae una réplica bootstrap del conjunto de entrenamiento inicial T= {(x_1, y_1), ..., (x_n , y_n)}, la probabilidad de que la observación i-ésima x_i (*i*=1, 2,..., *n*) sea incluida *m* veces (*m*= 0,1, 2,.., *n*) en esa muestra bootstrap, T^b, vendrá dada por la

distribución binomial B (*n*, *1/n*), donde *1/n* es la probabilidad que tiene x_i de ser seleccionada en cada extracción y *n* es el número de extracciones con reposición que se efectúan, en concreto la probabilidad de que x_i sea extraída m veces será:

$$P(m)=\binom{n}{m}\left(\frac{1}{n}\right)^{m}\left(1-\frac{1}{n}\right)^{n-m} \quad \textbf{(6.2)}$$

Cuando $1/n < 0'1$, es decir, para conjuntos de más de diez observaciones, se puede aproximar la distribución binomial a través de la Poisson, P(**8**), donde **8**=n(1/n)=1, y por tanto, la probabilidad de que x_i sea extraída *m* veces es:

$$P(m)=\frac{e^{-1}}{m!} \quad \textbf{(6.3)}$$

De este modo, cada observación tiene una probabilidad aproximada de *1/e* de no ser incluida (sustituyendo en la ecuación anterior *m*=0) en una réplica bootstrap. Por lo tanto, a la larga se puede esperar que, en término medio, aproximadamente el 37% de las observaciones se quedarán fuera de una muestra bootstrap. De esta forma, las posibles observaciones ruidosas del conjunto de entrenamiento no aparecerán en algunas de esas muestras. En ese caso, el clasificador construido bajo esas condiciones obtendrá mejor tasa de error que el construido en el conjunto de entrenamiento original con observaciones ruidosas y, lógicamente, debe tener una mayor influencia en la decisión final que otras versiones bootstrap. Así se explica que el clasificador obtenido por bagging consiga mejores resultados que los clasificadores individuales.

En realidad, los clasificadores construidos en las muestras bootstrap de los conjuntos de entrenamiento, unas veces obtienen un clasificador mejor que el original y en otras, uno peor que éste. El clasificador que sea superior al original tendrá una mayor probabilidad a posteriori y por ello, dominará en la decisión final. Por lo tanto, la combinación de las versiones bootstrap del clasificador permite obtener un clasificador mejor que el original.

Las diferentes versiones del bagging han sido estudiadas por varios investigadores, aunque quizás con más frecuencia en el ámbito de la regresión, que desde el punto de vista de la clasificación. En Breiman (1996) se muestra que bagging puede reducir el error, tanto de regresión como de clasificación, en los árboles de decisión[26]. Según este autor, bagging es beneficioso para clasificadores inestables únicamente, ya que principalmente lo que logra es reducir la varianza, mientras que para procedimientos estables puede incluso llegar a perjudicar el comportamiento del clasificador. Por ejemplo, para el método de clasificación del vecino más próximo.

Skurichina (2001) defiende que la estabilidad de los clasificadores lineales depende del tamaño del conjunto de entrenamiento. Por lo tanto, no se podría decir que bagging no es útil para un determinado clasificador de forma general, sino que dependerá de la aplicación concreta a la que se enfrente.

6.4. BOOSTING

Como ya se ha dicho, dado un conjunto de datos, un sistema de aprendizaje genera un clasificador capaz de predecir la clase de una nueva observación. La mayor o menor precisión de ese clasificador dependerá de la calidad del método utilizado, y de la dificultad que presente la aplicación concreta. Siempre que el clasificador obtenido supere a la regla por defecto, querrá decir que ha sido capaz de encontrar alguna estructura en los datos para conseguir esta ventaja. Boosting es un método que aumenta la precisión de un clasificador sacando provecho de su ventaja. De tal forma que utiliza el método de clasificación como una subrutina para producir un clasificador que consigue una alta precisión en el conjunto de entrenamiento.

---

[26] Breiman en su artículo "*Bagging Predictors*", utiliza para clasificación siete conjuntos (sólo uno de ellos es simulado), de la base de datos de la Universidad de California (salvo heart data), bagging consigue en promedio una la reducción del error frente a los árboles individuales de aproximadamente el 29%.Por otro lado, para regresión trabaja con cinco conjuntos, tres de los cuales son simulados, en este caso, la reducción del error que consigue bagging frente a los árboles individuales es de un 35% en promedio.

Boosting aplica el sistema de clasificación varias veces sobre el conjunto de entrenamiento, pero cada vez dirige la atención del aprendizaje a diferentes ejemplos del mismo. Una vez que el proceso ha terminado, los clasificadores básicos obtenidos se combinan en un único clasificador final que será muy preciso en el conjunto de entrenamiento. El clasificador final normalmente logra también una precisión elevada en el conjunto de test, según han demostrado diversos autores tanto teórica como empíricamente, entre otros Bauer y Kohavi (1999), Breiman (1998), Drucker y Cortes (1995), Dietterich (2000), Friedman, Hastie y Tibshirani (1998), Freund y Schapire (1996 y 1997), Ho (1998), Quinlan (1996) y Schapire (1999).

Aunque existen diversas versiones de algoritmos boosting la más extendida es la que proporcionan Freund y Schapire (1996) que se conoce como AdaBoost. Para simplificar se puede suponer que sólo existen dos clases y en el capítulo siguiente se generaliza a más de dos clases. Se parte del conjunto de entrenamiento T = $\{(x_1^b, y_1^b), ..., (x_n^b, y_n^b)\}$ donde Y toma en este caso los valores{-1, 1}. Se asigna a cada observación x_i el peso w_i^b, que inicialmente se iguala a 1/n, y posteriormente se irá actualizando en cada iteración. Se construye un clasificador básico a partir del conjunto de entrenamiento ponderado, que se representa por $C_b(x_i)$ y se aplica a cada uno de los ejemplos de entrenamiento. El error de ese clasificador se representa por ε_b y se calcula como

$$\varepsilon_b = \sum_{i=1}^{n} w_b(i)\,\xi_b(i) \quad donde \quad \xi_b(i) = \begin{cases} 0 & C_b(x_i) = y_i \\ 1 & C_b(x_i) \neq y_i \end{cases} \tag{6.4}$$

A partir del error del clasificador en la iteración *b*-ésima, se calcula la constante α_b, que se utiliza para la actualización de los pesos. En concreto estos autores hacen $\alpha_b = \ln(1 - \varepsilon_b / \varepsilon_b)$ y el nuevo peso para la iteración *b+1* será

$$w_{b+1}(i) = w_b(i)\ \exp(\alpha_b\ \xi_b(i)) \tag{6.5}$$

posteriormente se normalizan los pesos calculados para que la suma de todos ellos sea uno. Según Freund y Schapire, el error debe ser inferior al de la regla por defecto, $\varepsilon_b = 0{,}5 - \gamma_b$, donde γ_b representa la ventaja que obtiene el clasificador básico de la iteración

b-ésima sobre la regla por defecto, en el peor de los casos en que las dos clases tengan la misma probabilidad a priori, 0'5.

El siguiente ejemplo muestra cómo se actualizan los pesos en función del error cometido para la segunda iteración, *b*=2. Para ello, se han elegido dos valores próximos a los extremos del campo de variación de γ_b, que son 0 y 0,5. Si γ_b =0'499, entonces α_b = 0,004 y el nuevo peso para la segunda iteración será $w_i^2=1/n \exp(0{,}004 \cdot h_b(i))$. Luego, si la observación *i*-ésima está clasificada incorrectamente, su peso será $w_i^2=1/n \cdot 1'004$, mientras que, si ha sido clasificada correctamente, su peso en principio no se modifica, aunque al normalizar para que la suma de todos los pesos sumen uno, se verá disminuido. Si se considera ahora que γ_b =0'001, entonces α_b = 6,907 y el peso de una observación mal clasificada tendrá $w_i^2 =1/n \cdot 999$, mientras que, los pesos de las observaciones correctamente clasificadas se verán reducidos posteriormente en la normalización. La tabla 6.1 muestra cómo quedarían los pesos para la segunda iteración una vez normalizados, para un conjunto de 1000 observaciones.

n	peso inic	error	alfa	mal clasif	peso 2	peso 2 norm
1000	0,001	0,499	0,004	1	0,001004008	0,001002004
1000	0,001	0,499	0,004	0	0,001	0,000998004
1000	0,001	0,001	6,907	1	0,999	0,5
1000	0,001	0,001	6,907	0	0,001	0,000500501

Tabla 6.1 Ejemplo de cómo se actualizan los pesos en Adaboost.

Como se puede ver, aumenta la ponderación de las observaciones mal clasificadas y disminuye la de las clasificadas correctamente, forzando así al clasificador básico construido en la siguiente iteración a centrarse en aquellos casos que han resultado más difíciles. Además, las diferencias en la actualización son mayores cuando el error cometido por el clasificador básico es pequeño, porque si el clasificador consigue una precisión elevada se le da más importancia a los pocos fallos cometidos. Por tanto, la constante alfa puede considerarse como una tasa de aprendizaje calculada en función del error cometido en esa iteración. Además, esta constante también se utiliza en la regla

de decisión final, dando más importancia a los clasificadores básicos que cometen un menor error.

Este proceso se repite en todas las iteraciones desde *b* =1, 2, 3, ..,B. Para acabar, se construye el clasificador final como combinación lineal de los clasificadores básicos ponderados por la constante α_b correspondiente. En concreto será:

$$C(x)=sign\left(\sum_{b=1}^{B} \alpha_b\, C_b(x)\right) \qquad \textbf{(6.6)}$$

Algoritmo 6.2. AdaBoost (Freund & Schapire, 1996b).

1. Iniciar con $w_i^1 = 1/n$, i=1, 2, ..., *n*.

2. Repetir para b=1, 2, ...,B

   a) Construir el clasificador $C_b(x)\in\{-1,1\}$ utilizando los pesos w_i^b en T^b.

   b) Calcular: $\varepsilon_b=\sum_{i=1}^{n} w_i^b\,\xi_b(i)$ y $\alpha_b = \ln(1 - \varepsilon_b / \varepsilon_b)$

   c) Actualizar los pesos $w_i^{b+1}=w_i^b \exp(\alpha_b\, \xi_b(i))$ y normalizarlos.

3. Construir el clasificador final

$$C(x)=sign\left(\sum_{b=1}^{B} \alpha_b\, C_b(x)\right)$$

Adaboost puede aplicarse de dos maneras distintas, utilizando remuestreo o utilizando reponderación. En la versión que utiliza remuestreo, se obtiene el conjunto

de datos S_b para la iteración *b*-ésima, mediante una submuestra bootstrap extraída con reemplazamiento, utilizando como probabilidades para la extracción los pesos de las distintas observaciones en esa iteración. En la versión que utiliza la reponderación, el clasificador C_b tiene en cuenta directamente los pesos de los ejemplos. No existe una evidencia fuerte a favor de ninguno de los métodos frente al otro (Breiman,1998), (Freund y Schapire, 1997) y (Freund y Schapire, 1998)

6.4.1. Error de entrenamiento en Adaboost

Freund y Schapire demostraron que al aumentar el número de iteraciones B, el error de entrenamiento o aparente del clasificador combinado Adaboost tiende a cero a un ritmo exponencial. La regla de actualización de los pesos en el paso 2c del algoritmo 6.2, implica que

$$\sum_{i=1}^{n} w_{b+1}(i) = \sum_{i=1}^{n} w_b(i)\ \exp(\alpha_b\, \xi_b(i)) = \sum_{i=1}^{n} w_b(i) \left(\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\xi_b(i)} \leq$$

$$\langle\langle \alpha^r \leq 1-(1-\alpha)r \ \ para\ \alpha \geq 0\ y\ r \in [0,1] \rangle\rangle\ luego \qquad \textbf{(6.7)}$$

$$\leq \sum_{i=1}^{n} w_b(i) \left(1-\left(1-\frac{1-\varepsilon_b}{\varepsilon_b}\right)\xi_b(i)\right) = \left(\sum_{i=1}^{n} w_b(i)\right)\left(1-\left(1-\frac{1-\varepsilon_b}{\varepsilon_b}\right)\varepsilon_b\right)$$

combinando esta desigualdad sobre *b* = 1, 2, 3,...B, se tiene que

$$\sum_{i=1}^{n} w_{B+1}(i) \leq \prod_{b=1}^{B}\left(1-\left(1-\frac{1-\varepsilon_b}{\varepsilon_b}\right)\varepsilon_b\right) \qquad \textbf{(6.8)}$$

Por otro lado, la regla de decisión final se equivocará en la observación *i*-ésima si

$$\prod_{b=1}^{B}\left(\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\xi_b(i)} \geq \left(\prod_{b=1}^{B}\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\frac{1}{2}} \quad donde \quad \xi_b(i) = \begin{cases} 0 & C_b(x_i) = y_i \\ 1 & C_b(x_i) \neq y_i \end{cases} \qquad \textbf{(6..9)}$$

el peso final de cualquier instancia *i* es:

$$w_{B+1}(i)=w_1(i)\prod_{b=1}^{B}\left(\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\xi_b(i)} \tag{6.10}$$

Combinando las ecuaciones 6.9 y 6.10, se puede limitar inferiormente la suma de los pesos finales por la suma de los pesos finales de las observaciones donde la regla de decisión final se equivoca.

$$\begin{aligned}\sum_{i=1}^{n} w_{B+1}(i) &\geq \sum_{i:\, C_F(x)\neq y_i} w_{B+1}(i) \geq \left(\sum_{i:\, C_F(x)\neq y_i} w_1(i)\right)\left(\prod_{b=1}^{B}\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\frac{1}{2}} \\ &= \varepsilon_A\left(\prod_{b=1}^{B}\frac{1-\varepsilon_b}{\varepsilon_b}\right)^{\frac{1}{2}}\end{aligned} \tag{6.11}$$

donde ε_A es el error del clasificador final en el conjunto de aprendizaje, o error aparente. Combinando los resultados de 6.8 y 6.11 se tiene que:

$$\varepsilon_A \leq \prod_{b=1}^{B}\frac{1-\left(1-\frac{1-\varepsilon_b}{\varepsilon_b}\right)\varepsilon_b}{\sqrt{\frac{1-\varepsilon_b}{\varepsilon_b}}} = \prod_{b=1}^{B}\frac{1-(2\varepsilon_b-1)}{\sqrt{\frac{1-\varepsilon_b}{\varepsilon_b}}} = \prod_{b=1}^{B}\frac{2(1-\varepsilon_b)}{\sqrt{\frac{1-\varepsilon_b}{\varepsilon_b}}} = 2^B\prod_{b=1}^{B}\sqrt{\varepsilon_b(1-\varepsilon_b)} \tag{6.12}$$

que como $\varepsilon_b < 0'5$, entonces $\varepsilon_b(1-\varepsilon_b)<1/4$ y, por tanto, esta expresión muestra que a medida que aumenta el número de iteraciones (B), el error aparente tiende a cero a un ritmo exponencial.

6.4.2. La teoría del margen y Adaboost

Algunos experimentos con Adaboost han mostrado algo inesperado en el comportamiento de un sistema de clasificación, el error de test continúa disminuyendo al añadir más clasificadores en la combinación, incluso después de que el error de entrenamiento alcance el valor cero. Esto propicia un interés renovado en la búsqueda

de posibles explicaciones, y da origen a la conocida como teoría del margen (Schapire, Freund y otros, 1998).

El concepto del margen proviene de la teoría del aprendizaje estadístico y está relacionado con la dimensión de Vapnik-Chervonenkis (a la que normalmente se hace referencia como dimensión VC). De manera sencilla, puede decirse que la dimensión VC proporciona un límite superior en la precisión de modelos de clasificación[27]. Aunque el límite es amplio, se ha mostrado como una herramienta teórica importante en el reconocimiento de patrones y aprendizaje automático. El método de clasificación conocido como Máquina de Soporte Vectorial se basa en la teoría del aprendizaje estadístico y, más concretamente, en la idea de maximizar los márgenes. Intuitivamente el margen de una observación está relacionado con la certeza o confianza de su clasificación. Las observaciones para las que la clase asignada es correcta y con elevado grado de confianza, tendrán márgenes positivos y grandes. Los ejemplos con clasificación incierta o dudosa deben tener márgenes pequeños. Un margen pequeño es síntoma de inestabilidad en la clase asignada, es decir, el ejemplo puede ser asignado a clases distintas por clasificadores similares.

Para q clases el margen de un ejemplo x se calcula utilizando el grado de apoyo de las distintas clases $:_j(x)$, $j=1,2,..,q$ como

$$m(x) = \mu_k(x) - \max_{j \neq k} \mu_j(x) \qquad \textbf{(6.13)}$$

[27] La dimensión VC de un conjunto de funciones indicadores $\{f(x,"){0}\{0,1\}\}$, donde $"$ es un vector de parámetros y x 0 R^p, es igual al mayor número de vectores $x_1, x_2,.....,x_d$ que pueden ser separados en 2 clases diferentes para todas las 2^d formas distintas en que pueden ser etiquetadas utilizando este conjunto de funciones. Nótese, que si la dimensión VC es d, entonces existe como mínimo un conjunto de d puntos que puede ser fragmentados, pero en general no todos los conjuntos de d puntos podrán ser fragmentados.

donde k es la clase correcta de x y $\sum_{j=1}^{q} \mu_j(x) = 1$. Por tanto, todos los ejemplos erróneamente clasificados tendrán márgenes negativos y, aquellos clasificados correctamente, tendrán márgenes positivos.

Si se maximizan los márgenes (en inglés boosting the margins), se conseguirán clasificadores con una mayor confianza. Schapire y otros en su artículo de 1998 demostraron límites superiores para el error de test que dependen del margen. El principal resultado teórico para el problema dicotómico se da en el siguiente teorema.

Teorema. Sea , un espacio finito de clasificadores básicos[28]. Para todo *>0 y 2>0, con probabilidad al menos 1- * sobre la selección aleatoria del conjunto de entrenamiento **T**, cualquier combinación de clasificadores $C_F = \{C_1, C_2, ..., C_B\}$ f , utilizando la media ponderada satisface

$$P(error) \le P(margen\ \ entrenam\ \le \theta) + O\left(\frac{1}{\sqrt{n}}\left(\frac{\ln n \ln|\mathcal{H}|}{\theta^2} + \ln(1/\delta)\right)^{1/2}\right) \qquad \textbf{(6.14)}$$

donde P(error) es la probabilidad de que la combinación cometa un error al clasificar x0U^n, extraída aleatoriamente de la distribución del problema y P(margen entrenam #2) es la probabilidad de que el margen de un ejemplo extraído al azar del conjunto de entrenamiento no supere a 2. | , | es la cardinalidad de , .

Para el caso más general de , finito o infinito con dimensión VC *d*, se utiliza el siguiente límite, suponiendo que 1# *d* # *n*

[28] Un espacio finito de clasificadores básicos es, por ejemplo, el conjunto de todos los árboles de clasificación de un tamaño dado sobre un conjunto de variables discretas. Por ejemplo, el conjunto de todos los árboles monocorte sobre un conjunto de *p* variables dicotómicas contiene *p* elementos (clasificadores), uno para cada variable.

$$P(error) \leq P(margen\ \ entrenam\ \leq \theta) + O\left(\frac{1}{\sqrt{n}} \left(\frac{d \ln^2(n/d)}{\theta^2} + \ln(1/\delta) \right)^{1/2} \right) \qquad \textbf{(6.15)}$$

Ninguno de estos límites depende del número de clasificadores en la combinación. Aunque los límites son bastante holgados, muestran la tendencia de que márgenes grandes llevan a límites superiores más pequeños en el error de test.

6.4.3. Comparación de Bagging y Boosting

Los dos métodos construyen los clasificadores básicos en versiones modificadas del conjunto de entrenamiento y los combinan después en la regla de clasificación final. Sin embargo, se diferencian en el modo de obtener los clasificadores básicos. En concreto las tres principales diferencias son:

1. Todas las observaciones se utilizan en cada paso del boosting (no se aplica bootstraping), al menos en la versión inicial del mismo en Freund y Schapire (1996).

2.La regla de decisión final de Bagging no pondera a los clasificadores básicos de manera distinta según su precisión, mientras que en boosting si lo hace.

3. Por último, pero a la vez la más importante, el clasificador obtenido en cada paso del boosting depende de todos los anteriores, mientras que en Bagging son independientes.

6.5. BOSQUES ALEATORIOS

Como se señaló en el apartado 5.4., Krogh y Vedelsby (1995) afirman que la combinación ideal consiste en agregar clasificadores básicos de alta precisión y que estén el mayor número de veces posible en desacuerdo. Es decir, clasificadores básicos que presenten una tasa de error baja y estén poco relacionados entre sí.

Siguiendo esta idea, algunos métodos han intentado introducir aleatoriedad en el proceso de construcción de los clasificadores básicos. De esta manera se busca que disminuya la relación entre ellos, para aumentar la precisión del clasificador combinado. La aleatoriedad se puede introducir bien en el propio algoritmo de construcción del clasificador básico, o bien, en el conjunto de entrenamiento a utilizar para construirlo.

Cuando se utilizan árboles de clasificación como clasificadores base, hay varios métodos que generan vectores aleatorios que determinan el crecimiento de cada árbol de la combinación. El primero de éstos métodos es bagging (Breiman, 1996), donde cada árbol se desarrolla sobre una réplica bootstrap del conjunto de entrenamiento original. Es decir, se realiza una selección aleatoria con reemplazamiento de las observaciones en el conjunto de entrenamiento.

Otro ejemplo, es el método de selección aleatoria del corte (Dietterich, 1998), donde, en cada nodo, el corte se selecciona de manera aleatoria entre los *k* mejores cortes de todos los posibles. Es decir, en primer lugar se calcula la ganancia de información de todos los posibles cortes. A continuación, se ordenan y, de entre los *k* mejores, se elige uno al azar. El valor de *k* se fija previamente y es el mismo en todos los nodos. En tercer lugar, Ho (1998) plantea el método del subespacio aleatorio, que selecciona aleatoriamente un grupo de variables a utilizar en la construcción de cada árbol. Por último, Breiman (1999) propone un método que genera nuevos conjuntos de entrenamiento, mediante la aleatorización de las salidas en el conjunto de entrenamiento original.

El elemento común en todos estos métodos es que para el árbol *i*-ésimo, se genera un vector aleatorio $\mathbf{1}_i$, independiente de los vectores aleatorios utilizados para construir los árboles previos, ($\mathbf{1}_1$, $\mathbf{1}_2$, ...,$\mathbf{1}_{i-1}$), pero con la misma distribución. Por tanto, el árbol *i*-ésimo se construye utilizando el conjunto de entrenamiento y $\mathbf{1}_i$, obteniéndose un clasificador C(x, $\mathbf{1}_i$), donde x es el vector de características. Por ejemplo, en bagging, el vector aleatorio $\mathbf{1}$ se genera mediante la selección aleatoria, con reemplazamiento, de una muestra del mismo tamaño que el conjunto de entrenamiento original. Mientras

que, en la selección aleatoria del corte, $\mathbf{1}$ consiste en un vector de números enteros seleccionados aleatoria e independientemente entre 1 y *k*.

Después de generar un número suficientemente grande de árboles básicos, se recogen sus votos para ver cuál es la clase mayoritaria. Breiman (1999) llama a estos procedimientos *Bosques Aleatorios*, en inglés *Random Forest*.

En concreto, Breiman define un bosque aleatorio como *"un clasificador consistente en un conjunto de clasificadores con estructura de árbol* $\{C(x,\mathbf{1}_i); i=1,2,...\}$ *donde los* $\{\mathbf{1}_i\}$ *son vectores aleatorios independientes e idénticamente distribuidos y cada árbol recoge un voto unitario para la clase mayoritaria dada la observación x"*.

De esta definición hay que destacar dos aspectos que van a diferenciar los bosques aleatorios de boosting. En primer lugar, los vectores $\mathbf{1}_i$ son independientes e idénticamente distribuidos, mientras que en boosting el vector de los pesos en una iteración depende del comportamiento en las iteraciones anteriores. En segundo lugar, en un bosque aleatorio cada árbol aporta un voto unitario para la determinación de la clase mayoritaria, sin embargo, en boosting, el voto de cada árbol está ponderado por la constante $\mathbf{"}$, que se calcula a partir del error cometido por ese árbol.

Breiman propone dos maneras de construir los bosques aleatorios, ambas con el mismo objetivo, conseguir clasificadores básicos precisos y que, simultáneamente, estén lo menos relacionados entre sí que se pueda. La primera opción, construye los bosques aleatorios mediante la selección aleatoria de las variables de entrada, y la segunda, mediante combinaciones lineales aleatorias de estas variables.

Los bosques aleatorios con selección aleatoria de las variables de entrada suponen el uso simultáneo de bagging y de la selección aleatoria de variables. A partir del conjunto de datos original se realizan réplicas bootstrap, seleccionando con reemplazamiento el mismo número de elementos que hay en el conjunto de datos original. A continuación, se construye un árbol en cada uno de los nuevos conjuntos de

entrenamiento utilizando selección aleatoria de variables. Es decir, en cada nodo, se selecciona aleatoriamente un pequeño grupo de características entre las cuales se debe elegir la más adecuada para realizar el corte en ese nodo. El árbol se desarrolla al máximo y no se poda. El tamaño del grupo de variables seleccionado, *F*, se debe fijar previamente. Breiman(1999) prueba los valores F=1 y el primer número entero menor que $\log_2 p + 1$, donde p es el número de variables de entrada que caracterizan cada observación. Posteriormente, el mismo autor aconseja fijar el valor F en la raíz cuadrada de p, aunque según él, el procedimiento no es muy sensible al tamaño establecido.

En los experimentos que realiza en veinte conjuntos, frecuentemente utilizados en el aprendizaje automático, Breiman se muestra sorprendido de los buenos resultados obtenidos seleccionando una única variable ya que son, en general, sólo ligeramente inferiores a los logrados seleccionando un grupo, e incluso en ocasiones superior a éstos.

La selección aleatoria de variables agiliza el proceso, ya que se reduce el número de variables para las que hay que calcular la ganancia de información en cada nodo. Por tanto, la construcción de un bosque aleatorio de este modo será más rápida que la construcción de bagging. Además, es importante resaltar la diferencia entre la selección aleatoria de variables y la selección aleatoria del corte. En la primera, se elige al azar un pequeño grupo de variables y entre ellas se selecciona la que mejor corte produzca. En la selección aleatoria de corte, se calcula la ganancia de información que producen todas las variables en ese nodo, y posteriormente, se elige al azar un corte de entre los k mejores. Es decir, en la selección aleatoria de variables sólo hay que calcular la ganancia de información para aquellas variables que se seleccionan al azar, mientras que, en la selección aleatoria del corte, se calcula la ganancia de información para todas ellas. Para conjuntos de datos con muchas variables la diferencia de tiempo de cálculo puede ser importante.

Si el número disponible de características es pequeño, la selección aleatoria de variables haciendo F una porción considerable de p puede lograr una mejora en la

precisión de los clasificadores básicos, pero aumentará la relación entre ellos. Otra posibilidad consiste en generar nuevas variables partiendo de la combinación lineal aleatoria de un grupo de características. Se genera una variable especificando el número de variables que serán combinadas, L. En cada nodo, L variables se seleccionan aleatoriamente y se unen utilizando coeficientes que se obtienen de una distribución uniforme en el intervalo [-1,1]. De la misma manera, se generan F combinaciones lineales, y posteriormente, se realiza la búsqueda del mejor corte entre ellas. Este es el procedimiento del Bosque aleatorio con combinaciones lineales aleatorias de las variables. Breiman (1999) utiliza L=3 y F=2 u 8, siendo el valor de F elegido en función de la estimación del error. El valor L=3 permite que el número de posibles combinaciones sea suficiente como para que valores altos de F consigan una elevada precisión de los clasificadores básicos, pero sin provocar una relación muy fuerte entre los mismos.

Por tanto, a modo de resumen, se puede decir que los bosques aleatorios consisten en utilizar selección aleatoria de variables, o combinaciones aleatorias de ellas, en cada nodo, durante la construcción de los clasificadores básicos. Según Breiman, este tipo de procedimientos tiene algunas características deseables, como que su precisión es semejante a la de Adaboost, se construye más rápidamente que bagging y boosting, y se comporta bien cuando el poder discriminatorio está muy repartido entre muchas variables, lo que es una dificultad para muchos clasificadores individuales. Además, es relativamente robusto ante valores anómalos y la presencia de ruido.

La última característica, la robustez relativa ante observaciones ruidosas puede deberse a que Adaboost eleva el peso de los ejemplos mal clasificados. Las observaciones ruidosas, es decir, aquellas cuya etiqueta de clase es incorrecta, serán persistentemente mal clasificadas. Entonces, Adaboost se concentrará erróneamente en esas observaciones ruidosas y quedará distorsionado. Los bosques aleatorios no concentran ponderaciones en ningún subconjunto de los ejemplos y, por tanto, el efecto del ruido es menor.

6.5.1. Estimaciones out-of-bag

Tibshirani (1996) propone una estimación bootstrap del error de un clasificador de tipo bagging, a esta estimación Breiman (1999) la denomina estimación *"out-of-bag"* o fuera de la muestra. El objetivo es aprovechar las observaciones del conjunto de entrenamiento que no pasan a formar parte de cada una de las réplicas bootstrap utilizadas para construir los clasificadores básicos. Dado que estas observaciones no contribuyen a la formación de ese clasificador, pueden suponer un buen conjunto donde probarlo.

Se construyen B réplicas bootstrap (T_b) a partir del conjunto original (T) y en cada una se entrena un clasificador básico (C_b). Para cada observación x_i del conjunto original T, se agregan los votos de los clasificadores básicos donde $x_i \notin T_b$, es decir, aquellos clasificadores básicos que se han construido sin incluir x_i en su conjunto de entrenamiento. Este clasificador combinado es el clasificador "out-of-bag" o fuera de la muestra. La estimación "out-of-bag" para el error de generalización es la tasa de error del clasificador "out-of-bag" en el conjunto T.

En cada réplica bootstrap, aproximadamente el 37% de las observaciones se quedan fueran. Por tanto, las estimaciones out-of-bag están basadas en combinar aproximadamente una tercera parte de los clasificadores que utiliza la combinación principal en curso. Puesto que la tasa de error disminuye al aumentar el número de clasificadores combinados, la estimación out-of-bag tenderá a sobrestimar la tasa de error en la iteración actual. Para obtener estimaciones out-of-bag insesgadas, es necesario continuar el proceso más allá del punto donde la tasa de error de prueba converge. Por tanto, a diferencia de la validación cruzada, donde existe sesgo cuya cuantía es desconocida, las estimaciones out-of-bag son insesgadas.

Como señalan Wolpert y Macready (1996), las estimaciones mediante validación cruzada en las combinaciones de clasificadores que utilizan numerosas réplicas bootstrap exigen grandes esfuerzos computacionales. En cambio las estimaciones out-

of-bag son eficientes en este sentido, porque, se pueden calcular al mismo tiempo que se construye el clasificador combinado con un esfuerzo adicional muy pequeño. Breiman (1996) afirma que esas estimaciones, además, están cerca del óptimo, según las conclusiones que extrae de sus experimentos.

CAPÍTULO 7

EL MÉTODO BOOSTING Y SUS VARIACIONES

7.1. INTRODUCCIÓN

En los capítulos anteriores se ha establecido la base argumental de este trabajo. En primer lugar, se ha centrado el amplio tema dentro del que se engloba este trabajo, para ello se ha explicado algunos conceptos fundamentales de la clasificación. A continuación se han recogido algunos de los clasificadores individuales más utilizados, análisis discriminante, el vecino más próximo, árboles de clasificación y redes neuronales. Posteriormente, se han estudiado algunas características de los clasificadores como son el sesgo, la varianza y la inestabilidad de los mismos. Algunas de estas características ayudan a justificar el uso de combinaciones de clasificadores. También se ha incluido un breve comentario de la perspectiva bayesiana al respecto.

En el capítulo sexto se nombran algunas de las clasificaciones de los métodos de combinación de clasificadores, prestando especial atención a aquella que distingue entre métodos generadores y no generadores, según modifican o no los clasificadores individuales que combinan. Entre los métodos generadores, existen también diversos tipos: remuestreo, selección de variables, prueba y selección y métodos aleatorios de agregación. Es en el primero de estos grupos donde se encuadra el método boosting, objeto principal de este estudio y también el método bagging, que tiene algunos aspectos similares al boosting. En primer lugar, se estudia el método bagging que utiliza diversas muestras bootstrap del conjunto original para entrenar un grupo de clasificadores básicos que posteriormente combina por voto mayoritario.

A continuación, se analiza el método boosting centrándose especialmente en el algoritmo más utilizado de los de este tipo, el Adaboost. Este algoritmo aplica repetidamente el sistema de clasificación sobre el conjunto de entrenamiento, pero va dirigiendo la atención hacia los ejemplos más difíciles, y posteriormente combina los clasificadores construidos mediante voto mayoritario ponderado.

Para acabar el capítulo seis, se recoge el método de los bosques aleatorios, que genera un conjunto de árboles, introduciendo en su proceso de construcción cierto grado de aleatoriedad que asegure cierta diversidad en la posterior combinación

En este capítulo se profundiza un poco más en el estudio del método boosting que, como se ha dicho anteriormente, es el principal objetivo de este trabajo, se recogen primero los algoritmos que han dado lugar al desarrollo posterior de este método. También se exponen algunas de las modificaciones que se han propuesto al algoritmo Adaboost después de su divulgación, así como, la generalización para cuando existen más de dos clases y el estudio del tamaño adecuado de los árboles utilizados en la combinación. En concreto, el capítulo se estructura de la siguiente manera.

En primer lugar, se recogen cuales fueron los orígenes del boosting. Partiendo del algoritmo propuesto por Schapire en 1990 para potenciar la precisión de un clasificador

que sólo supera ligeramente a la regla del azar. El núcleo central de su algoritmo es la combinación de tres clasificadores básicos construidos sobre versiones modificadas del conjunto de entrenamiento, para forzar al clasificador hacia los ejemplos que se han clasificado erróneamente. Posteriormente, Freund desarrolló un algoritmo con la misma filosofía que el de Schapire, pero con una estructura algo más sencilla. Poco tiempo después ambos autores unieron esfuerzos para construir el algoritmo Adaboost, que es el más utilizado de todos ellos.

En el tercer apartado, se establece la relación entre el método boosting y los modelos aditivos generalizados. El método boosting se originó en el campo del Aprendizaje Automático y su comportamiento puede resultar, a priori, difícil de comprender para los estadísticos. Sin embargo, si se comprueba que boosting es un modelo aditivo generalizado, se le podrán aplicar las propiedades conocidas sobre éstos modelos, cuyo estudio está mucho más desarrollado en el ámbito estadístico.

A continuación, se exponen algunas de las modificaciones más interesantes a las que ha dado lugar el elevado interés y extenso trabajo desarrollado en este tema. Entre ellas, destaca la flexibilización del resultado de los clasificadores individuales a valores reales entre -1 y 1, dando lugar a lo que se conoce como Adaboost Real. Otra propuesta interesante es la utilización como función de pérdida del logaritmo de la verosimilitud, lo que da lugar a un nuevo algoritmo llamado Logitboost. También Breiman aporta una variación al algoritmo Adaboost, proponiendo el algoritmo Arcing, que simplifica la actualización de los pesos y la regla de combinación final.

El quinto apartado, aborda la generalización del método boosting para el caso en que existan más de dos clases. La opción más utilizada es convertir el problema con q clases en q problemas de dos clases. Esta es la idea en los algoritmos Adaboost.MH y Logitboost (q clases). Sin embargo, cuando los clasificadores básicos son capaces de discriminar entre más de dos clases, el algoritmo Adaboost, con una ligera modificación, es válido tanto en el caso dicotómico como en el caso general (Adaboost.M1).

Para cerrar el capítulo, se analiza el tamaño de los árboles utilizados como clasificadores básicos, planteando la conveniencia o no de dejar a los árboles que se expandan hasta minimizar la tasa de error como lo harían fuera del proceso de boosting, frente a la posibilidad de establecer un tamaño fijo para todos los árboles. En el caso de que se fije un tamaño para todos los árboles, ¿cuál debe ser?.

7.2. LOS ORÍGENES DEL BOOSTING

En el aprendizaje supervisado, a partir de un conjunto de entrenamiento dado, el sistema de clasificación generará una regla que asigna a cada observación una de las posibles clases. Esta regla será más o menos precisa dependiendo de la calidad del sistema de clasificación y de la dificultad del problema en cuestión. Intuitivamente, se puede ver que si el clasificador se comporta mejor que la regla por defecto, aunque sólo sea ligeramente, querrá decir que el sistema de clasificación ha encontrado cierta estructura en los datos para lograr su ventaja. Boosting es un método que potencia[29] la precisión del sistema de clasificación aprovechando su ventaja.

Boosting utiliza el sistema de clasificación como una subrutina para generar una regla de predicción, que tiene asegurada una elevada precisión en el conjunto de entrenamiento. Este método aplica el sistema de clasificación en sucesivas ocasiones sobre el conjunto de entrenamiento, pero cada vez centra la atención en ejemplos distintos.

Una vez finalizado el proceso, se combinan los clasificadores básicos en un único clasificador final muy preciso en el conjunto de entrenamiento. Además, este clasificador final, normalmente, es también muy preciso en el conjunto de prueba. Es decir, obtiene también notables resultados en la generalización, como han comprobado diversos autores tanto teórica como empíricamente.

[29] En realidad la traducción literal de Boosting es potenciando.

Aunque los orígenes pueden remontarse un poco más allá, donde se unen el campo del reconocimiento de patrones y la inteligencia artificial, generalmente se considera que el primer algoritmo de lo que hoy en día ya es una familia de algoritmos Boosting, fue el que propuso Robert E. Schapire en 1990 en su artículo "*The strength of weak learneability*". En ese artículo se propone un algoritmo para potenciar la precisión de lo que se denomina clasificador débil, aquél que es sólo ligeramente superior a la regla del azar o regla por defecto. Este algoritmo es adecuado, en principio, para el caso dicotómico, en el que sólo existen dos clases y se supone que el clasificador C_1 construido sobre el conjunto de entrenamiento T de tamaño *n*, es sólo ligeramente superior a la regla por defecto, es decir, su error es $\gamma = 0{,}5 - \varepsilon$, donde ε es la ventaja que el clasificador C_1 obtiene sobre la regla por defecto.

Para forzar al sistema de clasificación a aprender más sobre las observaciones difíciles de la distribución, se debe eliminar la ventaja del clasificador C_1 de alguna manera. Schapire propone, a partir del conjunto de entrenamiento T o T_1, construir un nuevo conjunto de entrenamiento T_2, del mismo tamaño *n*, forzando a que en este nuevo conjunto el clasificador C_1 no sea mejor que la regla por defecto, es decir, que la mitad de los ejemplos en T_2 estén bien clasificados por C_1 pero se equivoque en la otra mitad. Para lograr esto, se lanza una moneda al aire, si sale cara se extraen aleatoriamente ejemplos de T hasta encontrar uno que C_1 sea capaz de clasificar correctamente, es decir, $C_1(x_i) = y_i$. Y si sale cruz, se extraen ejemplos de T hasta que salga uno mal clasificado, o sea, $C_1(x_i) \ldots y_i$.

El autor explica que el tiempo de búsqueda es limitado, ya que si la precisión del clasificador C_1 está muy próxima a 0'5, el número de ejemplos extraídos hasta encontrar uno que cumpla con el requisito deseado, estar bien o mal clasificado según salga cara o cruz, sigue una distribución geométrica con probabilidad aproximadamente igual a 0,5. La esperanza de una distribución geométrica de parámetro p= 0'5, es $(1-p)/p = 1$. Por tanto, en promedio, se puede esperar que sólo sea necesario extraer dos observaciones, la deseada y otra.

Una vez formado el conjunto de entrenamiento T_2, se entrena de nuevo el sistema de clasificación pero utilizando, en esta ocasión, para ello el conjunto T_2, se obtiene así el clasificador C_2 ,cuyo error debe ser también al menos ligeramente inferior al de la regla por defecto. Por último, se crea un tercer conjunto T_3 , de tamaño *n*, eliminando de T aquellos ejemplos donde C_1 y C_2 coinciden. Es decir, se extraen observaciones de T hasta encontrar una en la que $C_1(x_i) \ldots C_2(x_i)$. Se entrena por tercera vez al sistema de clasificación creando el clasificador C_3 .

Para clasificar una nueva observación, x_i, si $C_1(x_i)=C_2(x_i)$ entonces se le asigna la clase acordada por ambos clasificadores, en los demás casos, C_F asigna $C_3(x_i)$. Es decir, en la combinación final C_F se aplica el voto mayoritario de C_1 , C_2 y C_3. Schapire demostró que el error del C_F está acotado por $g(\mathbf{g})= 3\mathbf{g}^2-2\mathbf{g}^3$, que es significativamente menor que el error original $\mathbf{g}$. Este proceso supone el núcleo fundamental del algoritmo boosting, y se utiliza reiteradamente para mejorar la precisión del clasificador final, llegando a formarse una estructura un tanto complicada que podría reflejarse en el siguiente gráfico.

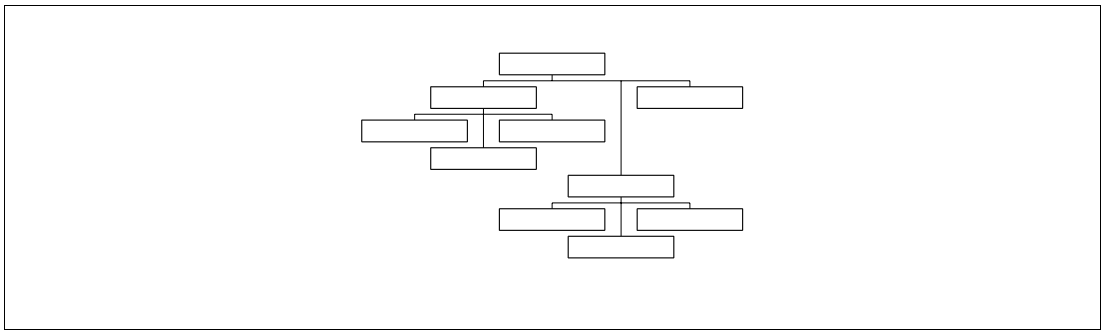


Figura 7.1. Estructura del algoritmo de Schapire (1990).

A pesar de la importancia que tuvo este primer algoritmo de Schapire su aplicación en la práctica resulta algo complicada. Además, la forma de construir los nuevos conjuntos es poco eficiente, debido a las llamadas recursivas que debe realizar. Poco tiempo después del trabajo de Schapire, Yoav Freund desarrolló un algoritmo más sencillo y eficiente que presentó en su artículo *"Boosting a weak learning algorithm by majority"* Este algoritmo también construye varias distribuciones diferentes sobre el

conjunto de observaciones, que se presentan al sistema de clasificación para centrar su atención en las regiones difíciles de la distribución desconocida. El sistema de clasificación genera un clasificador básico para cada distribución que recibe. Por tanto, los clasificadores básicos se comportarán bien en diferentes regiones del espacio de observaciones. El algoritmo boosting combina los clasificadores básicos en un clasificador final mediante el voto mayoritario simple.

En cada iteración, b=1,2, .., B, a los ejemplos del conjunto de entrenamiento se les asigna una distribución de probabilidad en función de la cual se genera el clasificador básico correspondiente a la iteración b-ésima. Aquellos ejemplos clasificados correctamente por ese clasificador reciben una marca. Después de las B iteraciones, los ejemplos que han sido marcados más de B/2 veces, son los ejemplos clasificados correctamente por el clasificador final, C_F. Las observaciones restantes, aquellas que han sido marcadas B/2 veces o menos[30], serán clasificadas incorrectamente por el clasificador final, el conjunto de estas observaciones se representa por L. Por tanto, el error de C_F en el conjunto de entrenamiento es |L| / |T|. El objetivo de este algoritmo boosting es minimizar este error. Para minimizar este error, Freund propone una estrategia de ponderación, que en cada iteración actualiza el peso de la observación x_i en la iteración b-ésima en función de b, B, γ y de cuántas veces x_i ha sido marcada antes de esa iteración. Donde, como ya se ha dicho, γ es la ventaja que los clasificadores básicos deben obtener como mínimo sobre la regla por defecto, siendo su error $\varepsilon_b = 0{,}5-\gamma$. Para más detalles sobre la estrategia concreta de actualización de los pesos en este algoritmo, véase Freund (1995).

Lo más importante del algoritmo de Freund es que consigue una estructura más sencilla que la que planteó Schapire originalmente, manteniendo e incluso mejorando la eficiencia, ya que consigue reducir el error de entrenamiento más rápidamente. La figura 7.2 ayuda a comprender mejor la diferencia de la estructura de ambos algoritmos.

[30] En este caso, se trabaja bajo el supuesto pesimista de que si los empates se rompen de forma aleatoria, la clase elegida es siempre errónea. Por ello, cuando una observación ha sido marcada exactamente B/2 veces, se considera que el clasificador final se equivoca.

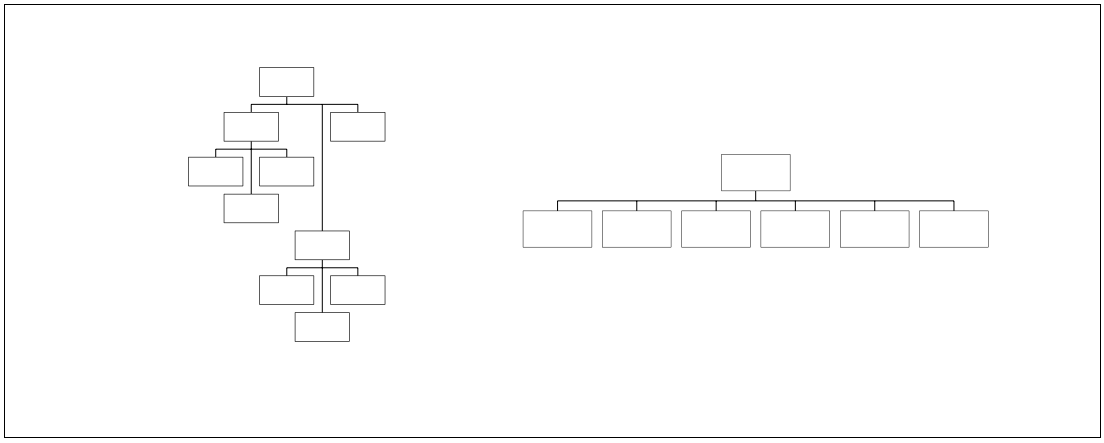


Figura 7.2. La parte izquierda del gráfico muestra la estructura compleja del algoritmo de Schapire (1990). A la derecha, la estructura más sencilla de Freund (1995).

Sin embargo, este algoritmo presenta también algunas deficiencias prácticas. En primer lugar, la regla de actualización de los pesos depende de γ, la ventaja que como mínimo deben obtener los clasificadores básicos sobre la regla por defecto. Es decir, esta es la ventaja que obtendrá en el peor de los casos, pero es posible que en realidad se obtenga una ventaja superior en muchos de los casos. En segundo lugar, Freund probó que el algoritmo boosting por mayoría requiere aproximadamente $1/\gamma^2$ iteraciones para reducir el error de entrenamiento hasta cero. Lo que quiere decir que, en el caso extremo de que γ=0,001, se necesitarán un millón de iteraciones. Durante las *B* iteraciones del proceso, algunos de los clasificadores básicos obtendrán un error menor que 0,5 - γ, pero este algoritmo no es capaz de aprovechar esta ventaja para acelerar el proceso.

Poco tiempo después, Freund y Schapire desarrollaron el algoritmo boosting más utilizado, el Adaboost (Adaptative boosting), que presentaron en el artículo *"Experiments with a new Boosting Algorithm"* publicado en 1996. En este caso, en la iteración *b*-ésima se aumenta el peso de los ejemplos mal clasificados por el clasificador básico correspondiente a esa iteración. Además, la regla de actualización de los pesos ya no depende de la constante γ, sino del error obtenido por el clasificador básico en esa iteración.

El comportamiento general de Adaboost es similar al de los otros algoritmos boosting, puesto que los clasificadores básicos se generan de forma sucesiva y se centra la atención en los ejemplos que resultan más difíciles. La principal diferencia entre Adaboost y el algoritmo de boosting por mayoría es la regla de actualización de los pesos. Adaboost utiliza una regla de actualización que depende del error del clasificador básico actual, no de la ventaja que deben conseguir como mínimo, **(**. Es decir, en Adaboost ese factor de la regla de actualización varía en cada iteración, mientras que en el otro caso permanece constante.

Además, otra diferencia es que a cada clasificador básico se le atribuye un peso **"**$_b$ en función del error que comete. Este peso se utiliza en la combinación final, puesto que Adaboost utiliza el voto mayoritario ponderado de los clasificadores básicos, en lugar del voto mayoritario simple. De esta forma, aquellos clasificadores básicos que consiguen una mayor precisión tendrán una mayor importancia en la combinación final.

El algoritmo Adaboost ha facilitado la aplicación del método boosting y desde su aparición han sido mucho los trabajos realizados al respecto, tanto empíricos comprobando sus buenos resultados, como teóricos, intentando explicar por qué se obtienen tan buenos resultados. Como resultado de esta extensa labor investigadora han surgido diversas modificaciones a este algoritmo, algunas de las cuales se recogen a continuación en el apartado 7.4.

7.3. RELACIÓN DEL MÉTODO BOOSTING CON LOS MODELOS ADITIVOS GENERALIZADOS

El método boosting se originó en el campo del Aprendizaje Automático y su comportamiento puede resultar al principio un poco oscuro y sorprendente para los estadísticos. Sin embargo, si se comprueba que Boosting es un modelo aditivo generalizado, se le podrán aplicar todas las propiedades conocidas sobre estos modelos, cuyo estudio está mucho más desarrollado en el ámbito estadístico.

Por tanto, el interés que presenta la relación entre el método Boosting y los modelos aditivos generalizados se basa fundamentalmente en que permite una mejor comprensión del funcionamiento y de los resultados que obtiene Boosting. En esta sección, en primer lugar, se va a exponer lo más brevemente posible en qué consisten los Modelos Aditivos Generalizados, que son ampliamente conocidos en el ámbito estadístico, para posteriormente resaltar la relación existente entre el método Boosting, para combinar clasificadores básicos, y estos modelos.

Los modelos más sencillos son los lineales, en estos modelos la salida se estima a partir de una combinación lineal de las *p* variables de entrada, de la siguiente forma,

$$\hat{y} = \alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots + \beta_p X_p \qquad \textbf{(7.1)}$$

A pesar de su interesante sencillez, los modelos lineales tradicionales no son adecuados en muchas situaciones, especialmente cuando se trabaja con datos reales. Por ello, se han desarrollado otros métodos más flexibles, como son los modelos aditivos generalizados, donde se sustituye cada término lineal por una forma funcional más general

$$\hat{y} = \alpha + \beta_1 f_1(X_1) + \beta_2 f_2(X_2) + \ldots\ldots + \beta_p f_p(X_p) \qquad \textbf{(7.2)}$$

Así, se consigue un modelo más flexible por las funciones $f_j(X_j)$, que en principio no tienen por qué ser paramétricas y se mantiene la aditividad permitiendo interpretar el modelo de manera muy similar al inicial. Las funciones f_j pueden ser iguales o no entre sí, pudiendo tomar formas lineales y no lineales. De manera aún más general, se pueden considerar modelos aditivos cuyos elementos son funciones de varias variables de entrada, o incluso de todas ellas. Normalmente, se utilizan funciones sencillas caracterizadas por un conjunto de parámetros y un multiplicador. Es decir, en este caso se sustituye la forma lineal $f(X) = \sum_{j=1}^{p} \beta_j X_j$ por el modelo

$$f_M(X) = \sum_{m=1}^{M} \beta_m \, b(X;\gamma_m) \qquad \textbf{(7.3)}$$

Estos modelos se conocen como expansión lineal de funciones base, donde β_m, m=1, 2,..., M son los coeficientes de expansión y b(X; γ_m): $R^p \to R$, se conocen como funciones base, y suelen ser funciones sencillas del conjunto de variables de entrada X, caracterizadas por un conjunto de parámetros γ_i. Cuando se utilizan como funciones base redes neuronales de una única capa oculta, estos parámetros determinan la combinación lineal de las variables de entrada. En el caso de los árboles de clasificación, estos parámetros son las variables de corte y los puntos donde éstos se realizan en los nodos internos, y las predicciones en los nodos hoja. La ventaja de estos modelos es que una vez las funciones base han sido determinadas, los modelos son lineales en estas nuevas variables, y el ajuste se realiza como antes.

Recordando la expresión (6.6) que da lugar al clasificador final de boosting, se puede ver que boosting supone una manera de ajustar una expansión aditiva de un conjunto de funciones base elementales. En el caso del boosting, las funciones base son los clasificadores individuales $C_b(x) \in \{-1,1\}$, que pueden ser redes neuronales, árboles de clasificación u otros clasificadores, y el clasificador final de boosting es

$$C(x)=sign\left(\sum_{b=1}^{B}\alpha_b C_b(x)\right) \qquad \textbf{(7.4)}$$

donde los coeficientes α_b actúan como coeficientes de expansión.

En muchas ocasiones los modelos aditivos se ajustan minimizando una función de pérdida promediada sobre el conjunto de entrenamiento, como puede ser la suma de cuadrados de los errores u otra función de pérdida basada en la verosimilitud

$$\min_{\{\beta_m,\gamma_m\}_1^M}\sum_{i=1}^{n}L(y_i,\sum_{m=1}^{M}\beta_m b(x_i,\gamma_m)) \qquad \textbf{(7.5)}$$

Para algunas funciones de pérdida L(y, *f*(X)) y/o funciones base b(X, γ), la resolución de esta expresión requiere un gran esfuerzo computacional para técnicas de optimización numérica intensivas.

Sin embargo, una alternativa algo más sencilla puede ser solucionar el problema, un poco más simple, de ajustar una única función base.

$$\min_{\beta,\gamma} \sum_{i=1}^{n} L(y_i, \beta b(x_i, \gamma)) \qquad \textbf{(7.6)}$$

La construcción de modelos aditivos paso a paso hacia delante aproxima la solución a la expresión (7.5), sumando progresivamente nuevas funciones base a la expansión sin ajustar los parámetros y los coeficientes de aquellas que se habían sumado previamente. En cada iteración *m*, se busca la función base óptima b(*x*, γ_m) y el correspondiente coeficiente β_m, que se debe sumar a la expansión existente en ese momento $f_{m-1}(x)$. De esta forma se obtiene $f_m(x)$, y se repite el proceso. Los términos añadidos con anterioridad no se modifican. El proceso se resume en el algoritmo 7.1.

Algoritmo 7.1. Modelización aditiva paso a paso hacia delante.

1. Iniciar con $f_0(x) = 0$
2. Repetir desde *m* =1 hasta M:
   a) Calcular $(\beta_m, \gamma_m) = \underset{\beta,\gamma}{arg\,mín} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma))$

   b) Actualizar $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

Si se considera el error cuadrático como función de pérdida

$$L(y, f(x)) = (y - f(x))^2 \qquad \textbf{(7.7)}$$

se obtiene

$$L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma)) = (y_i - f_{m-1}(x_i) - \beta b(x_i;\gamma))^2 = \\ = (r_{im} - \beta b(x_i;\gamma))^2 \qquad \textbf{(7.8)}$$

donde $r_{im} = y_i - f_{m-1}(x_i)$ recoge el error o residuo del modelo de la iteración *m-1* en la *i*-ésima observación. De esta forma, utilizando el cuadrado de los errores como función

de pérdida, en cada paso se añade a la expansión el término $\beta_m b(x; \gamma_m)$ que mejor ajusta los residuos que todavía existen en esa iteración. Sin embargo, el cuadrado de los errores no es una función de pérdida adecuada en los problemas de clasificación, por tanto, es necesario establecer otras funciones de pérdida.

Siguiendo con este razonamiento, boosting puede verse como un proceso equivalente a la modelización aditiva paso a paso hacia delante utilizando la siguiente función de pérdida exponencial.

$$L(y, f(x)) = \exp(-yf(x)) \tag{7.9}$$

Como se ha dicho anteriormente, para AdaBoost las funciones base son los clasificadores individuales $C_b(x) \in \{-1,1\}$. Utilizando la función de pérdida exponencial anterior, se debe resolver

$$(\beta_b, C_b) = \underset{\beta, C}{arg\,mín} \sum_{i=1}^{n} \exp[-y_i\,(f_{b-1}(x_i) + \beta_b C_b(x_i))] \tag{7.10}$$

para determinar el clasificador $C_b(x)$ y el correspondiente coeficiente β_b que se deben sumar en cada iteración. Esto se puede expresar como

$$(\beta_b, C_b) = \underset{\beta, C}{arg\,mín} \sum_{i=1}^{n} w_i^b \exp[-\beta_b\, y_i\, C_b(x_i)] \tag{7.11}$$

donde $w_i^b = \exp(-y_i f_{m-1}(x_i))$. Puesto que, w_i^b no depende ni de β_b ni de $C_b(x)$, puede considerarse como un peso que se aplica a cada observación. Además, este peso depende de $f_{m-1}(x_i)$ y, por tanto, los valores de los pesos individuales cambian en cada iteración.

La solución de la expresión (7.11) puede obtenerse en dos pasos. En primer lugar, para cualquier valor de $\beta > 0$, la solución de dicha expresión para $C_b(x)$ es

$$C_b = \underset{C}{arg\,mín} \sum_{i=1}^{n} w_i^b\, \xi_b(i) \quad donde \quad \xi_b(i) = \begin{cases} 0 & C_b(x_i) = y_i \\ 1 & C_b(x_i) \neq y_i \end{cases} \tag{7.12}$$

es decir, el clasificador que minimiza la tasa de error ponderada prediciendo el valor de *y*. Teniendo esto en cuenta, se puede escribir el criterio (7.11) como

$$e^{-\beta} \cdot \sum_{y_i = C_b(x_i)} w_i^b + e^{\beta} \cdot \sum_{y_i \neq C_b(x_i)} w_i^b \tag{7.13}$$

sabiendo además que $\sum_{i=1}^{n} w_i^b \xi_b(i) = \sum_{y_i \neq C_b(x_i)} w_i^b$, se puede sumar y restar la misma cantidad sin que cambie el resultado

$$\begin{gathered} e^{-\beta} \cdot \sum_{y_i = C_b(x_i)} w_i^b + e^{\beta} \cdot \sum_{y_i \neq C_b(x_i)} w_i^b - e^{-\beta} \cdot \sum_{y_i \neq C_b(x_i)} w_i^b + e^{-\beta} \cdot \sum_{y_i \neq C_b(x_i)} w_i^b \\ \Leftrightarrow (e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{n} w_i^b \xi_b(i) + e^{-\beta} \cdot \sum_{i=1}^{n} w_i^b \end{gathered} \tag{7.14}$$

Para minimizar esta expresión se deriva respecto de $ y se iguala a cero, con lo que se obtiene

$$\begin{gathered} (e^{\beta} + e^{-\beta}) \cdot \sum_{i=1}^{n} w_i^b \xi_b(i) - e^{-\beta} \cdot \sum_{i=1}^{n} w_i^b = 0 \Leftrightarrow \\ \frac{e^{\beta} + e^{-\beta}}{e^{-\beta}} \cdot \sum_{i=1}^{n} w_i^b \xi_b(i) = \sum_{i=1}^{n} w_i^b \Leftrightarrow \\ (e^{2\beta} + 1) \cdot \sum_{i=1}^{n} w_i^b \xi_b(i) = \sum_{i=1}^{n} w_i^b \Leftrightarrow \\ e^{2\beta} = \frac{\sum_{i=1}^{n} w_i^b - \sum_{i=1}^{n} w_i^b \xi_b(i)}{\sum_{i=1}^{n} w_i^b \xi_b(i)} = \frac{1 - \varepsilon_b}{\varepsilon_b} \Leftrightarrow \\ \beta = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_b}{\varepsilon_b} \right) \end{gathered} \tag{7.15}$$

donde $ es la tasa de error ponderada del clasificador construido en esa iteración.

$$\varepsilon_b = \frac{\sum_{i=1}^{n} w_i^b \xi_b(i)}{\sum_{i=1}^{n} w_i^b} \tag{7.16}$$

Una vez calculado el clasificador correspondiente a esa iteración y el coeficiente que le acompañará en la agregación, se actualiza la función *f(x)*, que será

$$f_b(x)=f_{b-1}(x)+\beta_b C_b(x) \tag{7.17}$$

y también, los pesos de la siguiente iteración, que pasan a ser

$$w_i^{b+1}=w_i^{b}\exp(-\beta_b y_i C_b(x_i)) \tag{7.18}$$

Es decir, en cada iteración se repite el mismo proceso, en primer lugar, se calcula el clasificador básico que minimiza la suma de los errores ponderados por los pesos correspondientes a esa iteración. Una vez entrenado el clasificador *b*-ésimo, se calcula su tasa de error ponderada y, a partir de ella, el coeficiente de expansión que acompañará al clasificador en la agregación. Como ya se ha dicho, en la modelización aditiva paso a paso hacia delante, estos clasificadores y sus coeficientes ya no se modifican posteriormente.

Puede verse que la actualización de los pesos en la ecuación (7.18), el coeficiente de expansión calculado en (7.15), y la función que se obtiene al final de este proceso son completamente equivalentes a las utilizadas en boosting. Por tanto, se puede concluir que boosting minimiza la función de pérdida exponencial, mediante un enfoque de modelización aditiva paso a paso hacia delante. Esto es importante, porque permite una mejor comprensión de este método que proviene del campo del aprendizaje automático y cuyo funcionamiento puede resultar, en principio, difícil de comprender en el ámbito estadístico.

Además, la condición impuesta en el primer paso de que el coeficiente de expansión tiene que ser positivo, **$**>0, se cumple en el caso de boosting, puesto que se exige que

todos los clasificadores básicos utilizados sean, al menos, clasificadores débiles, es decir, con un error ligeramente inferior a 0'5, como se demuestra a continuación.

$$\begin{aligned} &\beta>0 \Leftrightarrow \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)>0 \Leftrightarrow \\ &\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)>0 \Leftrightarrow \frac{1-\varepsilon}{\varepsilon}>1 \Leftrightarrow \\ &1-\varepsilon>\varepsilon \Leftrightarrow \frac{1}{2}>\varepsilon \end{aligned} \qquad \textbf{(7.19)}$$

7.4. VERSIONES DE ADABOOST

El algoritmo Adaboost descrito en el apartado 6.4, entrena varias veces al clasificador en el mismo conjunto de entrenamiento, pero centrando su atención en cada paso en los ejemplos que han sido mal clasificados en el paso anterior. Adaboost se limita al caso dicotómico y utiliza clasificadores básicos cuya salida puede tomar únicamente los valores -1 y 1, por ello se conoce también como Adaboost discreto, a partir del trabajo de Friedman y otros (2000). Desde su aparición en 1996, han sido varias las modificaciones propuestas para mejorar los algoritmos boosting, en este trabajo se recogen algunas de ellas.

Una generalización del Adaboost discreto fue propuesta en Freund y Schapire (1996b) y explicada en mayor profundidad en Schapire y Singer (1998), esta generalización utiliza predicciones con valores reales, que expresan además el grado de confianza, en lugar de los valores {-1, 1} que utiliza Adaboost. En este caso, los clasificadores básicos asignan un valor real entre -1 y 1. De tal forma que el signo de $C_b(x)$ informa de la clase predicha y el valor absoluto, $|C_b(x)|$ da una medida de la confianza en la predicción. El valor real de esta contribución se combina con las anteriores utilizando, como en el algoritmo original, un coeficiente α_b, aunque en esta ocasión se calcula de forma ligeramente distinta.

Friedman(2000) presenta una versión que llama Adaboost Real, donde los clasificadores básicos estiman la probabilidad de pertenencia a una determinada clase $p_b(x) = P_w(Y=1/X=x)0[0,1]$, donde el subíndice *w* indica que esa probabilidad se calcula sobre el conjunto ponderado utilizando los pesos w_i correspondientes a cada iteración. La contribución al clasificador final es la mitad de la transformación logit de esta probabilidad estimada.

Algoritmo 7.2. Adaboost Real.

1. Iniciar con $w_i^1 = 1/n$, i=1, 2, ..., *n*.
2. Repetir para b=1, 2, ...,B

a) Construir un clasificador para obtener una estimación de las probabilidades de las clases $p_m(x)=\hat{P}_w(y=1/x) \in [0,1]$ utilizando los pesos w_i^b en T^b.

b) Calcular: $f_b(x) = ½ \ln(p_m(x)/ (1-p_m(x)))$ 0 R.

c) Actualizar los pesos $w_i^{b+1}=w_i^b \exp(-y_i f_b(x_i))$ y normalizarlos.

3. Construir el clasificador final

$$C(x)=sign\left(\sum_{b=1}^{B} f_b(x)\right)$$

Como se ha visto en el apartado anterior, el algoritmo Adaboost minimiza una función de pérdida exponencial. Sin embargo, diversos autores defienden que para clasificación, en lugar de una función de pérdida exponencial, resulta más natural la elección como función de pérdida del logaritmo de la verosimilitud binomial. Basándose en esta función Friedman(2000) propone un nuevo algoritmo que llama Logitboost. Este nuevo algoritmo mantiene la misma filosofía de los anteriores algoritmos boosting, entrenar en sucesivas ocasiones al clasificador forzándolo a centrarse en los ejemplos difíciles, pero la forma de llevarlo a cabo es algo diferente.

En cada iteración se crea una función $f_b(x_i)$ que se ajuste lo mejor posible a z_i^b, que es la respuesta objetivo con la que se trabaja en la iteración *b*-ésima para la observación *i*-ésima. Esta respuesta se calcula en función de la verdadera clase de esa observación, que en este caso será {0,1}, de la probabilidad que se le ha asignado a la clase 1 en la iteración anterior, y del peso en esa iteración.

$$z_i^b = \frac{y_i - p_{b-1}(x_i)}{w_i^b} \qquad \textbf{(7.20)}$$

donde $p_{b-1}(x_i) = P_w(Y=1/X=x_i)$ y $w_i^b = p_{b-1}(x_i)\cdot(1-p_{b-1}(x_i))$

$F_b(x_i)$ es una función de agregación que se va actualizando en cada iteración y se calcula como $F_b(x_i) = F_{b-1}(x_i) + 0{,}5\, f_b(x_i)$ y a partir de ella se calcula la probabilidad de que la observación *i*-ésima pertenezca a la clase 1 en la iteración *b*.

$$p_b(x_i) = \frac{\exp(F_b(x_i))}{\exp(F_b(x_i)) + \exp(-F_b(x_i))} = \frac{1}{1 + \exp(-2F_b(x_i))} \qquad \textbf{(7.21)}$$

Para comprender mejor este algoritmo cada función de agregación, $F_b(x_i)$, puede considerarse como una estimación de la mitad del logaritmo neperiano del cociente de probabilidades de las clases.

$$F(x) \approx \frac{1}{2}\ln\left(\frac{p(x)}{1-p(x)}\right) \qquad \textbf{(7.22)}$$

Por tanto, Logitboost ajusta un modelo de regresión logística aditivo mediante optimización secuencial del logaritmo de la verosimilitud binomial, para más detalles véase Friedman(2000).

Una propiedad muy útil de este método es que produce directamente estimaciones de las probabilidades $\hat{P}[Y=1/X=x]$. Esto es muy importante para construir clasificadores cuando los costes de error en la clasificación no son iguales. Además, permite construir clasificadores con la opción de no asignar ninguna clase o asignar la etiqueta “duda” o “no clase” para ciertas observaciones. Una ventaja importante de Logitboost comparado

con métodos como las redes neuronales es que funciona bien sin necesidad de un ajuste afinado y sin llevar a cabo una sofisticada optimización no lineal, aunque esta ventaja es general para los algoritmos boosting.

Algoritmo 7.3. LogitBoost.

1. Iniciar con una función de agregación $F_0(x)$ /0 y probabilidades $p_0(x)$ /½, donde p(*x*) es la forma abreviada de $\hat{P}[Y=1/X=x]$

2. Repetir para b=1, 2, ...,B

a) Calcular la respuesta de trabajo y los pesos para i=1,2, ..,*n*

$$w_i^b = p_{b-1}(x_i)(1- p_{b-1}(x_i)) \; ; \quad z_i^b = \frac{y_i - p_{b-1}(x_i)}{w_i^b}$$

b) Ajustar una función $f_b(x_i)$ por mínimos cuadrados ponderados

$$f_b(x_i) = arg\min_f \sum_{i=1}^{n} w_i^b (z_i^b - f_b(x_i))^2$$

c) Actualizar la función de agregación y calcular las nuevas probabilidades.

$$F_b(x_i) = F_{b-1}(x_i) + 0{,}5\, f_b(x_i)$$

$$p_b(x_i) = (1+\exp(-2\, F_b(x_i)))^{-1}$$

3. Construir el clasificador final.

$$C(x_i) = \text{sign}(F_b(x_i))$$

Dada la definición de los pesos *w(x)* en aquellos ejemplos donde p(*x*) esté próximo a 0 ó a 1, *w(x)* llegará a ser muy pequeño. Esto puede causar problemas en el cálculo de $z(x_i)$, por lo que en Friedman(2000) se aconseja tomar las siguientes precauciones:

Si y=1, entonces calcular *z*= (y-*p*)/*p*(1-*p*) como 1/*p*. Como este cociente puede hacerse muy grande si *p* es pequeño, se debe limitar su valor en una cantidad determinada que se puede representar por *zmax*. El valor concreto elegido para *zmax* no

es crucial y Friedman(2000) afirma que valores comprendidos entre 2 y 4 funcionan bien. Por otro lado, si y=0, calcular z como -1/(1-*p*) con el límite inferior de -*zmax*.

Leo Breiman, autor del método Bagging (Breiman, 1996), mostró inmediatamente un gran interés por el método Boosting que Freund y Schapire propusieron casi simultáneamente a Bagging y que obtenía mejores resultados que éste. El considera que lo fundamental de esta técnica es el uso de remuestreo adaptativo y combinación, y por eso, los llama algoritmos Arcing[31]. La principal diferencia con el método Bagging consiste en entrenar los clasificadores básicos en muestras generadas a partir de distribuciones de probabilidad que van cambiando en función de los errores cometidos, en lugar de permanecer constantes como ocurre en Bagging. Esto implica que los clasificadores básicos utilizados en Bagging son independientes, mientras que en el caso de Boosting cada clasificador depende de los anteriores.

En Breiman (1996b) el autor explica que después de probar Adaboost empezó a sospechar que el éxito de éste no radicaba en su forma concreta sino en el remuestreo adaptativo que realiza, donde se aumentan los pesos de aquellos ejemplos que son clasificados erróneamente con mayor frecuencia. Para comprobarlo, probó tres maneras sencillas de actualizar los pesos o probabilidades. En todos ellos la actualización se realiza en función del valor $1 + m_i^h$, siendo m_i el número de veces que la observación *i*-ésima ha sido clasificada incorrectamente por los clasificadores básicos construidos hasta esa iteración. Breiman utilizó los valores h=1,2, 4 y éste último fue el que mejores resultados le proporcionó.Otra diferencia de arcing respecto a Boosting es que no utiliza ponderaciones en la regla de combinación final, lo que supone también una mayor sencillez del algoritmo al no tener que calcular y guardar esas ponderaciones.

[31] Arcing es el acrónimo del término inglés Adaptive Resampling and Combining. Breiman llama arc-fs al algoritmo Adaboost, en honor a Freund y Schapire, y a la modificación que él propone arc-x4, pero en este trabajo cuando se hable de arcing se referirá a arc-x4.

Algoritmo 7.4. Arcing.

1. Iniciar con $w_i^1 = 1/n$, i=1, 2, ..., *n*.

2. Repetir para b=1, 2, ...,B

a) Construir el clasificador $C_b(x)$ utilizando los pesos w_i^b para extraer con reemplazamiento la muestra aleatoria T^b.

b) Clasificar los ejemplos de T utilizando $C_b(x)$ y calcular $m_b(x_i)$ que es el número de errores cometidos por los *b* primeros clasificadores en la observación i-ésima.

c) Actualizar los pesos $w_i^{b+1} = \frac{1 + m_b(x_i)^4}{\sum_{i=1}^{n} (1 + m_b(x_i)^4)}$ y normalizarlos.

3. Construir el clasificador final.

$$C(x) = sign\left(\sum_{b=1}^{B} C_b(x) \right)$$

7.5. GENERALIZACIÓN DEL BOOSTING AL CASO DE *Q* CLASES

Hasta ahora, todos los algoritmos boosting que se han visto en este trabajo se basaban en el caso dicotómico, es decir, únicamente consideraban dos posibles clases. Sin embargo, aunque este supuesto facilita mucho la tarea y obtiene notables resultados, es en muchas ocasiones insuficiente en la realidad. La vida no es sólo blanco o negro sino que existen normalmente varios niveles de gris. En las aplicaciones económicas ocurre exactamente igual, como se puede ver en los siguientes ejemplos, a la hora de clasificar los clientes de una entidad financiera, además de distinguir entre morosos y no morosos, puede ser interesante establecer varios grupos de clientes según sean más o menos rentables para la entidad. En la predicción del fracaso empresarial, además de

discriminar entre fracasadas y no fracasadas, se pueden matizar niveles de fracaso o de éxito.

Por tanto, es necesario contemplar también aquellos problemas en los que las clases a predecir son más de dos. La opción elegida en la mayoría de los casos es modificar los algoritmos existentes para el caso dicotómico, generalizándolos para que sean capaces de trabajar con $q>2$ clases. En general, lo que se hace es transformar el problema con q clases en q problemas dicotómicos. En ocasiones se utilizan $q-1$ problemas dicotómicos quedando la q-ésima clase definida por eliminación, es decir, si hay q clases y ninguna de las q-1 primeras ha sido elegida, se asigna la clase q-ésima.

Ejemplos de este tipo de adaptación del caso dicotómico al caso más general de q clases son los algoritmos Adaboost.MH (Schapire y Singer, 1998) y LogitBoost para q clases (Friedman, 2000). En los problemas con q clases habrá que modificar la transformación logística, en este caso siendo $p_j(x)= P(y_j=1/x)$, se define la transformación logística múltiple simétrica como

$$F_j(x) = \ln p_j(x) - \frac{1}{q}\sum_{k=1}^{q} \ln p_k(x) \qquad \textbf{(7.23)}$$

Igualmente

$$p_j(x) = \frac{\exp(F_j(x))}{\sum_{k=1}^{q} \exp(F_k(x))}; \quad \sum_{k=1}^{q} F_k(x) = 0 \qquad \textbf{(7.24)}$$

La condición de centrado sirve para dar estabilidad numérica, haciendo que F_j tome un valor concreto, aunque si se le añade una constante a cada F_j las probabilidades continúan tomando el mismo valor.

Algoritmo 7.5. AdaBoost.MH

1. Transformar las *n* observaciones iniciales en *n x q* pares, {((x_i, 1), y_{i1}), ((x_i, 2), y_{i2}), ..., ((x_i, q), y_{iq})}, *i*=1, 2, ..., *n*. Donde y_{ij} es la respuesta {-1,1} para la clase *j* y observación *i*.

2. Aplicar Adaboost real al conjunto transformado, produciendo una función F(*x*, *j*) =$\mathbf{E}_b$ f_b (*x*, *j*).

3. Construir el clasificador final arg max$_j$ F(*x*, *j*).

Algoritmo 7.6. LogitBoost (*q* clases).

1. Iniciar con pesos w_{ij} =1/*n*, *i* =1,2, ..,*n* ; *j* =1,2,.., *q*, $F_j^0(x)$ / 0 y $p_j^0(x)$ / 1/*q* ∀*q*

2. Repetir para b=1, 2, ...,B

a) Repetir para j=1, 2, ...,*q*

i. Calcular la respuesta de trabajo y los pesos en la clase *j*-ésima

$$w_{ij}^{\ b} = p_j^{b-1}(x_i)(1- p_j^{b-1}(x_i)) \; ; \quad z_{ij}^{\ b} = \frac{y_{ij} - p_j^{b-1}(x_i)}{w_{ij}^{\ b}}$$

ii. Ajustar la función f_b por mínimos cuadrados ponderados

$$f_{bj}(x) = arg\min_f \sum_{i=1}^{n} w_{ij}^{\ b}(z_{ij}^{\ b} - f_{bj}(x_i))^2$$

b) Calcular

$$f_{bj}(x) = \frac{q-1}{q}\left(f_{bj}(x) - \frac{1}{q}\sum_{k=1}^{q} f_{bk}(x)\right), \quad F_j^{\ b}(x) = F_j^{b-1}(x) + f_{bj}(x)$$

$$p_j^{\ b}(x) = \frac{\exp(F_j^{\ b}(x))}{\sum_{k=1}^{q} \exp(F_k^{\ b}(x))}$$

3. Construir el clasificador final arg max$_j$ F_j (*x*).

El método más utilizado para construir los clasificadores básicos de boosting son los árboles de clasificación y éstos son capaces de discriminar entre más de dos clases. Por tanto, en este trabajo se propone el uso del algoritmo Adaboost.M1 que con solo unas ligeras modificaciones consigue adaptar Adaboost al caso general de *q* clases. Recordando el algoritmo 6.2, el clasificador básico construido durante la iteración *b*-ésima en el paso 2a, por ejemplo un árbol de clasificación, es capaz de distinguir entre más de dos clases, ya que en este caso no se le fuerza a distinguir únicamente entre {-1,1}. En el paso 2b, se calcula el error del clasificador de esa iteración sumando los pesos de los ejemplos mal clasificados, es decir, aquellos en los que su clase real no coincide con la clase predicha por el clasificador básico de esa iteración, sean cuales sean de entre las *q* clases posibles.

Algoritmo 7.7. AdaBoost.M1.

1. Iniciar con $w_i^1 = 1/n$, i=1, 2, ..., *n*.
2. Repetir para b=1, 2, ...,B
   a) Construir el clasificador $C_b(x)$ utilizando los pesos w_i^b en T^b.
   b) Calcular: $\varepsilon_b = \sum_{i=1}^{n} w_i^b \xi_b(i)$ y $\alpha_b = \frac{1}{2} \ln(1 - \varepsilon_b / \varepsilon_b)$
   c) Actualizar los pesos $w_i^{b+1} = w_i^b \exp(\alpha_b \xi_b(i))$ y normalizarlos.
3. Construir el clasificador final

$$C(x) = \arg\max_{y_j} \sum_{b=1}^{B} \alpha_b \delta(C_b(x), y_j)$$

El clasificador final utilizaba el signo de la suma ponderada de los resultados de los clasificadores individuales. Ahora el clasificador final, ante una nueva observación, recogerá para cada clase, la suma ponderada de los votos que ha recibido en esa clase de los B clasificadores básicos. Por tanto, la clase predicha será la que haya obtenido un mayor valor en el voto ponderado. Esta expresión es válida tanto para el caso

dicotómico como para el caso general. Además, en el caso dicotómico la expresión 7.25 es equivalente a la expresión utilizada en el algoritmo 6.2.

$$C(x)=arg\max_{y_j} \sum_{b=1}^{B} \alpha_b \delta(C_b(x), y_j) = arg\max_{y_j} \sum_{b:C_b(x)=y_j} \alpha_b \qquad \textbf{(7.25)}$$

Tal y como se hizo en el capítulo 6, puede verse un ejemplo de cómo se actualizan los pesos en función del error cometido para la segunda iteración, *b*=2. Para ello se han elegido dos valores próximos a los extremos del campo de variación de γ_b, que son 0 y 0,5. Si γ_b =0'499, entonces ε_b = 0,002 y el nuevo peso para la segunda iteración será $w_i^2=1/n$ exp (-0,002 $\eta_b(i)$). Luego, si la observación *i*-ésima está clasificada incorrectamente su peso será w_i^2 =1/*n* A1'002, mientras que, si ha sido clasificada correctamente, su peso en principio no se modifica, aunque al normalizar para que la suma de todos los pesos sumen uno, se verá disminuido. Si se considera ahora que γ_b =0'001, entonces ε_b = 3,453 y el peso de una observación mal clasificada tendrá $w_i^2=1/n$ A31'6, mientras que, los pesos de las observaciones correctamente clasificadas se verán reducidos posteriormente en la normalización. La siguiente tabla muestra cómo quedarían los pesos para la segunda iteración una vez normalizados, para un conjunto de 1000 observaciones.

n	peso inic	error	alfa	mal clasif	peso 2	peso 2 norm
1000	0,001	0,499	0,002	1	0,001002002	0,001001002
1000	0,001	0,499	0,002	0	0,001	0,000999002
1000	0,001	0,001	3,453	1	0,031606961	0,030668298
1000	0,001	0,001	3,453	0	0,001	0,000970302

7.6. EL TAMAÑO DE LOS ÁRBOLES EN BOOSTING

En la mayoría de las aplicaciones del boosting, el clasificador básico se construye siguiendo el mismo procedimiento que cuando este clasificador se utiliza en solitario, la única diferencia es que el método boosting lo llama repetidamente durante las iteraciones. La forma de actuar del clasificador básico es la misma que utilizaría si estuviese en cualquier otro contexto, con los mismos datos y los mismos pesos. Es decir, no se tiene en cuenta el hecho de que el modelo final es una combinación lineal de un gran número de esos clasificadores. Por ejemplo, cuando se utilizan árboles de clasificación, generalmente se utilizan los mismos algoritmos de desarrollo y poda del árbol. Sólo en algunas ocasiones se realizan pequeñas modificaciones, como obviar la poda, para facilitar la programación y acelerar el proceso.

Sin embargo, si se considera boosting como un modelo aditivo tal y como se ha visto en el apartado 7.3, ese proceso secuencial puede no ser el más adecuado en algunas situaciones. Naturalmente el objetivo del clasificador es estimar, lo más precisamente posible, la verdadera función que determina la distribución de los datos, que se representará por $D(x)$. Pero en boosting este objetivo debe cumplirlo el modelo aditivo final, no los clasificadores básicos de manera individual. Por ejemplo, si la verdadera distribución de los datos fuese aproximadamente una combinación lineal de las p variables originales, entonces aplicando boosting a árboles de un sólo corte y dos nodos hoja, que de ahora en adelante en este trabajo se llamarán árboles simples o monocorte, se conseguiría la mejor aproximación a la verdadera distribución.

Construir árboles de mayor tamaño sería una complicación innecesaria, y podría llevar a un aumento de la tasa de error, porque el resultado implicaría interacciones de orden superior entre las características. Los árboles mayores optimizarían las tasas de error de los clasificadores básicos a modo individual, para los pesos de esa iteración, e incluso producirían menores tasas de error global en las primeras iteraciones. Pero, después de suficientes repeticiones el modelo basado en árboles simples conseguiría un mejor resultado.

De manera más formal, se puede considerar una expansión de la verdadera distribución de los datos en el marco de la descomposición ANOVA (Análisis de la Varianza) de una función propuesta por Friedman (Friedman, 1991).

$$D(x)=\sum_{j} f_j(x_j)+\sum_{j,k} f_{jk}(x_j,x_k)+\sum_{j,k,l} f_{jkl}(x_j,x_k,x_l) \qquad \textbf{(7.26)}$$

La primera suma representa la función más próxima a *D(x)* que es aditiva en las variables originales, en análisis de la varianza esto se conoce como los efectos principales. La segunda representa la mejor aproximación utilizando interacciones entre dos variables, dados los efectos principales anteriores. La tercera suma recoge interacciones entre tres variables, y así sucesivamente.

Si la verdadera distribución *D(x)* puede aproximarse correctamente por una expansión de ese tipo detenida a un grado de interacción bajo, entonces permitir al clasificador básico que produzca interacciones de orden superior puede disminuir la precisión del modelo boosting final. Esto se debe a que generalmente los modelos más complejos se comportan peor a la hora de generalizar, porque tienen un mayor riesgo de haberse sobreajustado al conjunto de entrenamiento. Si se utilizan árboles de clasificación, las interacciones de nivel más alto las producen los árboles de mayor tamaño, mayor profundidad o mayor número de nodos hoja.

Por tanto, en aquellas situaciones donde la verdadera distribución admita una descomposición ANOVA de bajo orden, se puede aprovechar esta estructura para mejorar la precisión limitando el tamaño de los árboles construidos en las distintas iteraciones. Este tamaño no debe ser muy superior al verdadero nivel de interacción en *D(x)*. Lamentablemente, esta distribución subyacente es desconocida en la mayoría de los problemas, especialmente en los problemas reales, por tanto el tamaño adecuado de los árboles básicos se puede considerar como un parámetro a fijar en el procedimiento, o bien, se puede estimar su valor óptimo mediante alguna técnica de selección de modelos, como es la validación cruzada.

El tamaño de un árbol se puede limitar mediante el procedimiento habitual de poda. Se parte del árbol más largo posible y se obliga a eliminar tantos cortes como sean necesarios hasta conseguir el tamaño deseado. Esto puede suponer una pérdida de tiempo importante cuando el tamaño buscado es pequeño. El tiempo empleado en construir el árbol es proporcional al tamaño del árbol mayor antes de empezar a podar. Por tanto, se puede conseguir un importante ahorro computacional y una mayor sencillez del proceso, deteniendo el desarrollo del árbol al llegar al tamaño máximo, o de manera similar a un número máximo de nodos hoja. Las razones habituales a favor de construir árboles grandes y podarlos posteriormente pierden su sentido en el contexto del boosting ya que los posibles defectos de un árbol individual se pueden compensar por el resto de árboles construidos en el proceso del boosting.

Puesto que se va a detener el proceso en un número de nodos hoja o tamaño determinado es importante fijar el modo de elección de los cortes. La elección más habitual es seleccionar en cada nodo aquel corte que obtenga una mayor ganancia de información, o equivalentemente el que mayor reducción de la heterogeneidad consiga. Cada corte dicotómico aumenta el número de nodos hoja en uno, de tal forma que se continúa haciendo cortes hasta alcanzar el número máximo de nodos hoja establecido.

El tamaño máximo, M, se aplica a todos los árboles en las B iteraciones del proceso boosting. Por tanto, este valor es un parámetro a definir en el procedimiento boosting, como lo es el número de iteraciones. Si se desea encontrar el valor óptimo para un conjunto de datos en particular, se puede estimar mediante alguna técnica de selección de modelos, como puede ser el uso de la validación cruzada para minimizar la tasa de error. Si se utiliza el número de nodos hoja, M, para limitar el tamaño de los árboles y se hace M=2 se consideran únicamente los efectos principales, sin tener en cuenta nigún tipo de interaccíón. Si por el contrario se hace M=3 se estarán recogiendo también las interacciones de primer orden, y así sucesivamente. En Friedman (2001) se afirma que en base a la experiencia valores de M entre 4 y 8 obtienen buenos resultados, siendo además estos resultados poco sensibles a la elección de uno u otro valor en ese rango. Por tanto, se puede ajustar el valor de M probando varios de ellos y eligiendo aquel que

obtenga una menor tasa de error en un conjunto de validación. Sin embargo, raras veces se obtienen mejoras significativas frente al uso de M•6. Estos autores llaman al uso de boosting con árboles de tamaño fijo, *árboles logísticos aditivos* y utilizan el acrónimo ALT de Additive Logistic Trees.

A la hora de comparar este modelo con aquellos en los que no se limita el tamaño de los árboles, si sus tasas de error son similares, hay que tener en cuenta que este modelo consigue un ahorro computacional, lo que supone una ventaja importante en aquellos conjuntos de datos especialmente grandes donde el tiempo de computación se convierte en un problema. Además, conviene recordar una vez más la preferencia de los modelos más sencillos frente a los más complejos cuando las tasas de error son similares, debido a la mejor capacidad de generalización.

Otra ventaja de las aproximaciones con interacciones de bajo orden es la visualización del modelo. En concreto, para modelos aditivos en las variables de entrada la contribución de cada variable, x_j , j=1, 2, ..., p puede verse en un gráfico que representa x_j frente a la suma de todos los árboles simples que utilizan esta variable. Mientras que los modelos con un mayor nivel de interacciones son más difíciles de interpretar. Si las interacciones son únicamente entre dos variables la contribución se puede representar mediante gráficos de contorno o malla en perspectiva (tridimensional). Aunque los modelos sin interacción no consigan la mejor precisión, pueden resultar útiles con carácter descriptivo para la interpretación del modelo resultante.

TERCERA PARTE

CAPÍTULO 8

PREDICCIÓN DEL FRACASO EMPRESARIAL

8.1. INTRODUCCIÓN

En economía, cualquier persona, empresa u organización que establezca alguna relación con una entidad, por ejemplo, como inversor, concesión de crédito o accionista, está interesado en analizar el comportamiento y viabilidad de la empresa en cuestión. Los investigadores económicos han estudiado esta cuestión desde diversos puntos de vista considerando las diferentes formas de fracaso financiero, incluyendo suspensión de pagos, insolvencia y quiebra.

Esencialmente, el término quiebra hace referencia al cese de la actividad de la empresa presentando una solicitud de declaración de quiebra, debido a problemas financieros severos de la empresa para afrontar las obligaciones financieras con sus acreedores. Por el contrario, las otras formas de fracaso empresarial no necesariamente conducirán al final de la actividad de la empresa. Una descripción más detallada sobre las distintas formas de fracaso empresarial puede encontrarse en Calvo-Flores y García (2002) y Zopounidis y Dimitras (1998).

El principal objetivo de la predicción del fracaso empresarial es diferenciar las empresas que tienen una elevada probabilidad de fracasar de las empresas sanas. Este es, en principio, un problema de clasificación dicotómico. Sin embargo, a menudo se incluyen otros grupos que pueden dar mayor flexibilidad al análisis. Este grupo intermedio puede incluir empresas para las que es difícil realizar una predicción clara. Otra posibilidad es distinguir diferentes niveles de fracaso, o bien, considerar aquellas empresas fracasadas que consiguen sobrevivir finalmente mediante planes de reestructuración, incluyendo fusiones y adquisiciones (Theodossiu et al, 1996).

La clasificación de las empresas en grupos en función de su riesgo de fracaso se realiza habitualmente a partir de sus características financieras, utilizando información derivada de los estados financieros disponibles (balance de situación y cuenta de pérdidas y ganancias).

Los ratios financieros calculados a través de las cuentas de los estados contables son los criterios más utilizados en la predicción del fracaso empresarial. Sin embargo, la predicción del fracaso únicamente en función de esos ratios financieros ha sido criticada por varios autores (Dimitras et al., 1996; Laitinen, 1992). Las críticas se han centrado principalmente en que los ratios financieros sólo son los síntomas de los problemas financieros y de funcionamiento que una empresa atraviesa, pero no la causa de estos problemas. Para superar estas deficiencias, algunos autores han señalado la importancia de considerar información cualitativa adicional para la predicción del fracaso. Esta información cualitativa incluye criterios tales como la gestión de las empresas, su organización, el nicho de mercado al que pertenece la empresa y la posición que ocupa en él, las ventajas competitivas que posee, etc. (Zopounidis, 1987). Sin embargo, esta información no está disponible al público y en consecuencia es muy difícil poder recogerla. Esta dificultad explica que la mayoría de los estudios en predicción de fracaso se basen sólo en ratios financieros.

Una amplia revisión de los trabajos más relevantes en el problema de predicción del fracaso puede encontrarse en los libros de Altman (1993) y Zopounidis y Dimitras

(1998). Una referencia fundamental en este campo es el trabajo de Altman (1968) donde se utiliza análisis discriminante lineal para desarrollar modelos de predicción del fracaso. Altman utiliza discriminante lineal para construir un modelo de predicción de fracaso, considerando varios ratios financieros por primera vez en un contexto multivariante. Este trabajo, pionero junto con el de Beaver(1966) aunque éste a nivel univariante, motivó a otros investigadores hacia el uso de técnicas estadísticas para estos propósitos, durante las décadas de los 70 y los 80.

Desde mediados de los 80 nuevas técnicas no paramétricas han centrado el interés de los investigadores en este campo, entre otras, sistemas expertos, aprendizaje automático y redes neuronales. Los resultados de estos estudios han mostrado que estas nuevas técnicas son apropiadas para la predicción del fracaso, proporcionando resultados satisfactorios en comparación a las técnicas estadísticas tradicionales.

8.2. ESTADO ACTUAL DE LA PREVISIÓN DEL FRACASO EMPRESARIAL

La predicción del fracaso empresarial es importante por la gran cantidad de individuos o entidades que se pueden ver afectadas por este hecho, tanto desde el interior de la empresa, como son trabajadores y accionistas, como desde fuera de ésta, inversores y analistas financieros, entidades financieras, clientes, proveedores y otros acreedores, auditores e instituciones públicas como la Agencia Tributaria o la Seguridad Social.

Además, dependiendo de la importancia de la empresa para el sector y para la zona o zonas donde esté ubicada (municipio, región, país) las repercusiones pueden ampliarse a otros niveles. Hoy en día no cabe ninguna duda de que la globalización económica es una realidad, por tanto, se puede suponer que también las repercusiones pueden llegar a ser globales. Ante este panorama las distintas administraciones públicas (locales, regionales, nacionales e incluso internacionales) deben interesarse por prever el fracaso de determinadas empresas.

Aunque existe un consenso generalizado al afirmar la importancia de predecir el fracaso empresarial, paradójicamente no existe un mismo grado de acuerdo a la hora de definir qué es el fracaso empresarial, o en otras palabras, cuándo fracasa una empresa. Si se considera, desde una perspectiva global, el fracaso como la no consecución de los objetivos planteados, se puede afirmar que una empresa habrá fracasado cuando no haya conseguido sus objetivos, especialmente aquellos referentes a la rentabilidad, solvencia y supervivencia.

Un problema común en todos los estudios empíricos sobre fracaso empresarial es la limitación de los datos disponibles a la hora de fijar qué empresas han fracasado y cuáles no. En la mayoría de los casos, se distingue entre empresas quebradas y empresas sanas o, como mucho, se consideran también las empresas que han entrado en un proceso concursal de suspensión de pagos. Según Lizárraga (1996) esto se debe a que, de esta forma, se consigue trabajar con un concepto riguroso, ajeno a interpretaciones subjetivas y presente en bases de datos asequibles, lo que supone una mayor objetividad para esa investigación empírica.

En la aplicación práctica realizada se pretende ser un poco más ambicioso porque se va a considerar la posibilidad de distinguir varios tipos de fracaso, considerando para ello como empresas fracasadas a aquellas empresas que han desaparecido (disueltas) y aquellas otras que han sido absorbidas por otras empresas. Aún siendo conscientes de que no en todos los casos la absorción es un síntoma de fracaso, se ha considerado como tal pues rompe la continuidad de la empresa en sí y porque, además, no se dispone de información para discernir en qué casos la absorción es el resultado de una mala gestión, y cuándo se debe al elevado atractivo de la empresa por su boyante situación.

La propia dificultad que implica la elección de la definición de fracaso empresarial que se va a utilizar, produce una merma en la capacidad predictiva del modelo, porque en ocasiones las firmas que estando en situación crítica son vendidas aparecen como empresas sanas y no como fracasadas. Mientras que otras que consiguen sobrevivir tras atravesar una suspensión de pagos aparecen como fracasadas.

Para prevenir el fracaso empresarial es fundamental conocer las causas por las que puede fracasar una empresa. En este sentido Argenti (1976) cita las siguientes como posibles causas del fracaso entre otras: deficiencias en la administración, fallos en los sistemas de contabilidad, incapacidad para adaptarse a cambios en el entorno, embarcarse en proyectos que escapan a las posibilidades reales de la empresa, excesivo o inadecuado uso del endeudamiento como vía de financiación, el riesgo inevitable en el mundo empresarial, etc.

En esta misma línea, Gabás (1997) realiza una clasificación de esas causas según su origen esté en el exterior o en el interior de la compañía y un tercer grupo que denomina causas especiales. Esta clasificación puede verse en la tabla 8.1.

1.- Causas con origen externo:

, De mercado:

- Competencia excesiva.
- Fuerte caída de la demanda.

, Del entorno económico-social:

- Fase recesiva del ciclo económico.
- Crisis sobrevenida (Crisis del petróleo, Guerra de Irak, conflictos locales, etc.)
- Política económica del gobierno.
- Cambios sociales radicales y significativos.

2.- Causas con origen interno:

- Ineficacia de la dirección.
- Estrategias erróneas o inadecuadas.
- Sistema productivo ineficiente.
- Inversiones improductivas.

• Excesivo endeudamiento, agravado en ciertas épocas con altos tipos de interés. • Final del ciclo de vida del producto. • Fracaso de empresas del grupo. • Problemas concursales no resueltos. • Alta morosidad.
3.- Causas especiales: • Nuevas empresas: presentan una tasa de mortalidad muy elevada en los primeros años de actividad.

Tabla 8.1. Causas del fracaso empresarial, según Gabás (1997)

En realidad, generalmente, es muy difícil conocer cuáles son las causas del fracaso cuando se analiza un conjunto de empresas, por lo que se ha de trabajar con los síntomas que resultan visibles. En la mayoría de los trabajos sobre fracaso empresarial se considera aceptable el estudio de los ratios financieros y, más concretamente, de su deterioro, como síntomas del mismo. Si bien, algunas compañías que atraviesan dificultades caen en la tentación de manipular la información contable.

8.2.1. Los trabajos pioneros

Un tema en el que no hay dudas en la literatura sobre fracaso empresarial es que los trabajos pioneros en este ámbito fueron los de Beaver (1966) y Altman (1968). Beaver fue uno de los primeros en usar técnicas estadísticas para analizar ratios financieros en la predicción del fracaso empresarial. En aquel trabajo utilizó 30 ratios medidos en 79 compañías entre fracasadas y no fracasadas, de los 30 ratios iniciales se seleccionaron 6 como significativos. La información recogía los últimos 5 años anteriores al fracaso. El análisis consistía en comparar las medias de cada ratio en ambos grupos (fracasadas y no fracasadas). En aquel trabajo Beaver construyó, en cada ratio y para cada año por separado, un clasificador dicotómico con el objetivo de minimizar el error en la clasificación.

A pesar de la importancia innegable del trabajo de Beaver, parece obvio que en una cuestión tan compleja como la predicción del fracaso empresarial, los modelos univariantes no pueden ser capaces de recoger toda la dimensión del problema. Por el contrario, la información necesaria para conocer la posición financiera de una empresa es claramente multivariante. Se trata de valorar las distintas facetas financieras desde una visión global del conjunto. Para lograr este objetivo un gran número de investigadores en este campo, que se citan más adelante, han utilizado técnicas estadísticas de análisis multivariante para acometer la labor de predecir el fracaso empresarial.

El primero en aplicar las técnicas estadísticas multivariantes fue Altman en 1968. En concreto, aplicó Análisis Discriminante Lineal utilizando el Z-score, para medir la proximidad de una compañía al fracaso. Utiliza una muestra con dos clases distintas, fracasadas y sanas, emparejadas por tamaño y sector (para eliminar la influencia de estas variables en los resultados), que recoge información de los dos años previos al fracaso.

Los resultados que obtuvo causaron una gran sorpresa, con la combinación lineal de 5 ratios consigue una tasa de acierto superior al 90% en la muestra de prueba si utiliza la información del último año anterior al fracaso, y por encima del 70% cuando trabaja con la información de dos años antes del fracaso. Esto es algo habitual en todos los trabajos de este tipo, a medida que se alejan del año del fracaso es más difícil predecirlo y, por tanto, disminuye el porcentaje de empresas bien clasificadas.

8.2.2. Superación del Análisis Discriminante Lineal

Hasta finales del siglo XX el trabajo de Altman siguió siendo la referencia, aplicando el modelo experimental a otros períodos, sectores y zonas geográficas, utilizando técnicas estadísticas, variables, horizontes temporales y diseños muestrales idénticos o con pequeñas modificaciones. Todos estos trabajos, utilizan muestras de empresas fracasadas y sanas, y se consideran aceptables aquellos modelos que logran niveles de acierto clasificatorio similares a los de Altman. La elección de las variables a utilizar

para la clasificación se aborda desde una perspectiva principalmente empírica, en función de la información disponible y del mayor o menor acierto en la predicción. Entre estos trabajos pueden citarse, Deakin (1972), Blum (1974), Edmister (1972), Libby(1975), Scott (1981) y Taffler (1982).

A pesar de su uso generalizado el Análisis Discriminante Lineal presenta algunos problemas entre los que cabe destacar dos, en primer lugar, la distribución de las variables explicativas debería ser normal multivariante para que el análisis discriminante funcione correctamente. Pues bien, en la práctica los ratios financieros no presentan casi nunca distribuciones normales. En segundo lugar, las funciones más utilizadas son las funciones lineales, que necesitan que las matrices de covarianzas que recogen el grado de homogeneidad de los grupos sean iguales. Cuando estas matrices no son iguales, lo que sucede frecuentemente, deberían emplearse funciones cuadráticas en lugar de funciones lineales. Martín (1997) recoge una descripción más detallada de los problemas del análisis discriminante.

Para intentar superar las dificultades que presenta el análisis discriminante se desarrollaron modelos basados en la probabilidad condicional, *logit* y *probit*, que parten de supuestos notablemente más flexibles. El primer trabajo sobre predicción del fracaso empresarial utilizando modelos basados en la probabilidad condicional fue desarrollado por Ohlson (1980). Aunque sus resultados fueron modestos, constituyó el germen del que surgieron otros trabajos entre los que cabe citar Mensah (1983), Zavgren (1985), Casey y Bartczak (1985) y Peel y Peel (1988).

A partir de entonces se han ido incorporando otras técnicas útiles para cualquier problema de clasificación, tales como los árboles de clasificación, las redes neuronales y máquinas de soporte vectorial o los sistemas basados en reglas. En Somoza (2002) puede verse una interesante clasificación de los trabajos relacionados con la predicción del fracaso empresarial, aunque no se trata de un estudio pormenorizado sino de una revisión de las aportaciones más fundamentales en este campo de investigación. Para ello utiliza diversos criterios que pueden dar lugar a las siguientes agrupaciones:

1. Objetivo del estudio. Lo que se pretende con cada estudio, así en la mayoría de los casos se elabora un modelo de predicción (Beaver (1966), Altman (1968), Deakin (1972), Blum (1974) y otros), en ocasiones se comparan modelos para elegir entre las mejores alternativas (Elam (1975), Hamer (1983), Frydman et al. (1985)), o bien una formulación teórica que sustente la selección de los resultados (Wilcox (1971)).

2. Definición de fracaso. En la mayoría se utiliza como subrogado de ésta la quiebra legal (Altman (1968), Deakin (1972), Zavgren (1985)), en el caso español la suspensión de pagos o la quiebra (Lizárraga (1995) o Gallego et al. (1997)), también se utiliza la morosidad a una entidad de crédito (Beaver (1966) o Edminster (1972)).

3. Variables utilizadas. Las más usuales son los ratios financiero-contables extraídos a partir de las cuentas anuales depositadas por las empresas, ya sea desde un enfoque univariante (Beaver (1966)) o multivariante (Altman (1968), Deakin (1972), entre otros). También han sido utilizados ratios y sus desviaciones, así como en valores relativos (Edminster (1972), Blum (1974), Altman et al. (1977)). Además, en ocasiones se incluyen variables cualitativas, como pueden ser la actividad principal, el retraso o no en la presentación de las cuentas anuales, la rotación o no de los administradores (Peel y Peel (1988)); si la edad de la empresa supera un determinado umbral, si el número de administradores supera a la media muestral, si se ha realizado una ampliación de capital en un período determinado (Keasey y Watson (1987)); si la empresa presenta beneficios (Ohlson (1980)); reducción de dividendos en el ejercicio anterior al analizado (Flagg et al. (1991)). Por otro lado, la inclusión de variables externas como el deflactor del PIB (Ohlson (1980)) o los valores medios de algunos ratios para los sectores utilizados (Platt y Platt (1990)).

4. Técnica aplicada. Las más utilizadas han sido las estadísticas y dentro de éstas el análisis discriminante multivariante (Altman (1968), Blum(1974), entre otros). Así mismo, y en parte para subsanar las ya citadas limitaciones del discriminante, se han utilizado los modelos de probabilidad condicional, el logit (Ohlson(1980), Casey y Bartzack (1984)) y probit (Zmijewski (1984)). Más recientemente, han aparecido

investigaciones que han utilizado técnicas como las particiones iterativas (Frydman et al. (1985), Gabás (1990)) y el empleo de redes neuronales artificiales (Serrano Cinca (1993), Koh et al. (1999), Barney et al. (1999), Rodríguez López (2002)).

5. Resultados obtenidos. Hay dos tipos de trabajos, unos que sólo aplican los resultados a la muestra que sirvió para la construcción del modelo (Beaver (1966), Altman et al. (1974)) y aquellos otros en que se valida sobre una segunda muestra distinta de la de entrenamiento, que puede ser contemporánea a la inicial, aunque a veces se recomienda que sea posterior (Altman (1968), Deakin (1972), Zmijewski (1984)).

A continuación se presenta una tabla a modo de resumen

1. Objetivo del estudio.	- Elaborar un modelo predictivo: Beaver (1966), Altman (1968), Deakin (1972), Blum (1974) y otros. - Comparar modelos: Elam, (1975), Hamer (1983), Frydman et al. (1985) - Una formulación teórica: Wilcox (1971).
2. Definición de fracaso.	- Quiebra legal: Altman (1968), Deakin (1972), Zavgren (1985). - Suspensión de pagos o la quiebra: Lizárraga (1995) o Gallego et al. (1997). - Morosidad a una entidad de crédito Beaver (1966) o Edminster (1972).

3. Variables utilizadas.	- Ratios financiero-contables: Beaver (1966), Altman (1968), Deakin (1972). - Ratios y sus desviaciones o en valores relativos: Edminster (1972), Blum (1974), Altman et al. (1977). - Variables cualitativas: Peel y Peel (1988), Keasey y Watson (1987), Ohlson (1980), Flagg et al. (1991). -Variables externas: Ohlson (1980), Platt y Platt (1990).
4. Técnica aplicada.	- Análisis discriminante: Altman (1968), Blum(1974). - Modelos de probabilidad condicional, el logit (Ohlson(1980), Casey y Bartzack (1984)) y probit (Zmijewski (1984)). - Particiones iterativas: Frydman et al. (1985), Gabás (1990). - Redes neuronales artificiales: Serrano Cinca (1993), Koh et al. (1999), Barney et al. (1999), Rodríguez López (2002).
5. Resultados obtenidos.	- Se aplican a la muestra entrenamiento: Beaver (1966), Altman et al. (1974). - Se valida en muestra distinta: Altman (1968), Deakin (1972), Zmijewski (1984).

Tabla 8.2. Clasificación de trabajos de predicción de fracaso empresarial, Somoza (2002).

8.2.3. Los ratios financieros utilizados

Todos estos trabajos tienen algunos rasgos en común, uno de ellos es el uso de ratios financieros calculados a partir de la información recogida en los estados contables, Balance de Situación y Cuenta de Pérdidas y Ganancias. Se parte de un grupo de ratios, en la mayoría de los casos elegidos en base a tres criterios:

1. Haber sido utilizado frecuentemente en la literatura existente de predicción del fracaso empresarial.

2. Disponibilidad de la información. En muchos casos los estudios empíricos deben ceñirse a la información que existe disponible ante la imposibilidad de realizar una recogida de información propia debido, entre otras cosas, al elevado coste que supondría.

3. Elección propia del autor o autores del estudio. En base a su experiencia ciertos autores ignoran algunos ratios e incluyen otros que consideran oportunos.

En Gabás (1997) se recoge un resumen de los ratios que fueron destacados como más importantes en algunos estudios, que pueden verse en la tabla 8.3.

1. Beaver (1966)	1. Cash-flow/ Total obligaciones.
2. Altman (1968)	1. Fondo de maniobra/Activo total. 2. Resultados acumulados/A. total. 3. Resultados A.I..I./A. total.
3. Dambolena y Khoury (1980)	1. Resultados A.I..I./A. total. 2. Resultados A.I..I./Intereses pagados. 3. Resultados acumulados/A. total.
4. Mensah (1983)	1. Cash-flow/ Total obligaciones. 2. Resultado ordinario A.I./Ventas.
5. Altman (1983): Sector ferrocarriles	1. Cash-flow/Gastos fijos. 2. Gastos operativos/Ingresos operativos. 3. Resultados brutos/A. total.
6. Holder (1984): Sector Transporte	1. Realizable y disponible/A. total. 2. Financiación básica/A. total. 3. F. maniobra/ Ventas.
7. Gentry, Newbold y Whitford (1985)	1. Ratios de cash-flow.
8. Laffarga et al. (1985)	1. Beneficio neto antes de imptos/A. total. 2. B.N.A.I./Pasivo exigible.
(1986)	1. B.N.A.I./A. total. 2. B.N.A.I./P. exigible. 3. Créditos más cartera de valores/A. total. 4. P. exigible/P. total. 5. B.N.A.I./F. propios.
(1987)	1. B.N.A.I./A. total. 2. A. fijo/A. total. 3. Reservas/P. total.

9. Pina (1989)	1. A. circulante/A. total. 2. A. circulante menos tesorería/A. total. 3. Beneficio neto/F. propios.
10. Gabás (1990)	1. Rtdo ejer A. Imptos/A. total. 2. G. financieros/F. ajenos menos inmovilizado en curso 3. Provisiones más amortiz/Inmov. bruto. 4. Dividendos/ F. propios. 5. Tesorería/ Exigible a corto plazo. 6. Provisión I.S.S./ Rtdo ejer A. Imptos.
11. Mora (1994)	1 Rentabilidad. 2. Liquidez.
12. Lizárraga (1996)	1. Rtdo neto/ A. total. 2. F. propios/Deuda total. 3. A. defensivos/Ventas.
13. Arqués (1997)	1. Rentabilidad financiera. 2. Liquidez a corto plazo. 3. Carga financiera.

Tabla 8.3. Comparación de los ratios más importantes en algunos trabajos, Gabás (1997).

Según Gabás, la variable más común en estos estudios es la de rentabilidad. En las demás existe una elevada variabilidad reflejo probablemente de la heterogeneidad de los datos de cada estudio, ya que recogen información referente a empresas de sectores diversos, tamaños variados, información que proviene de periodos temporales distintos, así como de distintos lugares geográficos, ya sea de regiones dentro de un mismo país o entre países.

8.3. CRITICAS Y VENTAJAS DE LOS MODELOS DE PREDICCIÓN DEL FRACASO EMPRESARIAL

A pesar de la gran cantidad de literatura que existe en la predicción del fracaso empresarial, lo que demuestra, sin duda alguna, el interés de la cuestión, este tipo de trabajos han recibido una serie de críticas que no deben restar importancia a la tarea desarrollada sino, por el contrario, servir de estímulo para el perfeccionamiento de los modelos existentes.

1. La falta de estabilidad de los modelos. Se obtienen modelos distintos para los diferentes sectores, tamaños de empresas diversos, localización geográfica (regiones o países) y períodos temporales.

Los modelos funcionan mejor para muestras homogéneas de empresas respecto al sector o tipo de actividad o de industria. Así mismo, los síntomas del fracaso pueden ser diferentes en las grandes empresas que en las PYMES, por ello en muchos trabajos se emparejan las empresas fracasadas y sanas, por tamaño, generalmente utilizando el activo total o el número de empleados como testigos de la dimensión de la empresa. Además, las normas contables y mercantiles suelen variar de un país a otro, dificultando el uso de un mismo modelo en países distintos. Del mismo modo, los sistemas de clasificación entrenados para un período de tiempo, tampoco se pueden extrapolar sin más a períodos distintos debido a posibles cambios en las normativas o en el ciclo económico.

2. En ocasiones los resultados son brillantes en el conjunto de aprendizaje, pero modestos en conjuntos de prueba que no han participado en el entrenamiento del clasificador. Este problema se conoce como *sobreaprendizaje o sobreajuste,* y puede ser un síntoma de que los modelos se dejen engañar por relaciones casuales o fortuitas presentes en el conjunto de aprendizaje, pero no fuera de él. De esta forma, se pierde la utilidad del sistema de clasificación al no poder ser utilizado para la predicción en nuevas observaciones.

3. Son incapaces de detectar la posible manipulación de la información contable por algunas empresas que atraviesan momentos delicados que pueden conducir al fracaso. Estos modelos se adecuan a situaciones límites de fracaso (quiebra, suspensiones de pago) y no a estados previos del mismo.

4. No son capaces de recoger los distintos procesos que pueden llevar al fracaso financiero de una empresa, Laitinen (1993).

5. En muchos estudios el clasificador construido es uno más entre los muchos posibles, quizá sólo con una ligera ventaja en el acierto clasificatorio.

A pesar de todas estas críticas siguen existiendo razones suficientemente sólidas a favor de la utilidad de los modelos de predicción del fracaso empresarial, como las que menciona Martín (1997).

1. En el caso del Análisis Discriminante se obtiene una puntuación única que permite una *ordenación de las empresas analizadas*, a la vez que la discriminación dicotómica entre empresas sanas y empresas en peligro de fracasar.

2. Estos modelos, cualquiera que sea el que se utiliza en concreto, pueden considerarse como *un instrumento más a disposición del analista* a la hora de dictaminar la salud financiera de una empresa o su posible acercamiento al fracaso.

3. El *uso de estos sistemas desde el interior de la empresa* por los órganos encargados de su gestión puede ayudar a prever situaciones de riesgo y tomar las medidas oportunas para intentar evitarlo.

4. El nuevo marco regulatorio de Basilea II supone un gran cambio respecto al Acuerdo de Capital de Basilea de 1988, y su posterior modificación de 1996, porque permite que las entidades de crédito basen los cálculos de sus *requerimientos mínimos de capital por riesgo de crédito en sus sistemas internos de clasificación de acreditados*

y la medición de los factores de riesgo asociados a cada categoría. Para ello, dichas entidades deberán demostrar al supervisor que cuentan con procedimientos internos suficientes para valorar la adecuación de su capital en relación con su perfil de riesgo.

Este nuevo enfoque supone retos importantes para las entidades y los supervisores. Las entidades de crédito que quieran usar modelos internos deberán prepararse para cumplir con los requisitos establecidos. En concreto, deberán evaluar si son adecuados sus sistemas de medición, bases de datos y procedimientos para cumplir con tales requerimientos. Por otro lado, los supervisores deberán estar preparados para validar los modelos internos utilizados en el cálculo de los requisitos mínimos de capital y valorar si la entidad cuenta con capital suficiente en relación a su perfil de riesgo.

8.4. ANÁLISIS DESCRIPTIVO DE LAS ESTADÍSTICAS SOBRE SUSPENSIONES DE PAGOS, QUIEBRAS, FUSIONES Y DISOLUCIONES

Antes de abordar la aplicación práctica de este trabajo conviene analizar, aunque sólo sea desde un ámbito descriptivo, la realidad de los conceptos que se van a utilizar en España, para ello se ha recurrido a las estadísticas que proporciona el Instituto Nacional de Estadística (INE) al respecto. En concreto, se ha trabajado con la Estadística de Suspensiones de Pagos y Declaraciones de Quiebra y con la Estadística de Sociedades Mercantiles que recogen esa información.

La Estadística de Suspensiones de Pago y Declaraciones de Quiebra (SQ) recoge información sobre el número de expedientes de suspensiones de pagos y declaraciones de quiebra que se inician en los Juzgados de Primera Instancia e Instrucción en el territorio nacional, incluidas Ceuta y Melilla. El ámbito poblacional de la estadística lo constituyen todos los expedientes de suspensiones de pagos y declaraciones de quiebra iniciados en los Juzgados. El mes de referencia es aquél en que se inicia el expediente, la estadística tiene periodicidad trimestral desde 2002 y la información que proporciona parte del primer trimestre de 1990.

La fuente de información son los Juzgados de Primera Instancia y los Juzgados de Primera Instancia e Instrucción, que remiten los datos al INE en un cuestionario normalizado diseñado conjuntamente por el Consejo General del Poder Judicial y el INE. Se obtienen datos por provincias y por comunidades autónomas del número de suspensiones y quiebras, así como del activo y pasivo de las empresas que están en esas situaciones. También se clasifican por actividad económica y naturaleza jurídica de la empresa, la clase de quiebra, la causa de la suspensión y la proposición de pago.

La Estadística de Sociedades Mercantiles se creó por Orden de 30 de septiembre de 1938. La Ley de Reforma y Adaptación de la Legislación Mercantil a las directivas comunitarias, el Real Decreto Legislativo de 22 de diciembre de 1989, sobre el texto refundido de la Ley de Sociedades Anónimas y el posterior Reglamento del Registro Mercantil, afectaron a la parte operativa y conceptual de esta estadística a partir de enero de 1990.

Su objetivo es medir la demografía de las sociedades. Esta estadística ofrece información mensual de las sociedades creadas, de las sociedades disueltas y de aquellas en las que se ha producido modificaciones de capital. Se obtienen datos por provincias, comunidades autónomas y total nacional, a partir de los datos suministrados por el Registro Mercantil Central que recoge información de todo el territorio nacional, incluidas Ceuta y Melilla.

8.4.1. Suspensiones de pagos

TABLA 8.4 EMPRESAS EN SUSPENSIÓN DE PAGOS EN ESPAÑA.				
	1 TRIM.	**2 TRIM.**	**3 TRIM.**	**4TRIM.**
1990	70	63	97	121
1991	191	220	176	211
1992	261	252	265	357
1993	397	359	343	347
1994	336	246	181	206
1995	214	146	157	133
1996	184	172	145	148
1997	135	121	102	121
1998	94	86	78	90
1999	69	80	64	77
2000	92	80	55	79
2001	97	73	58	86
2002	87	119	90	112
2003	106	103	71	87
2004	72	89	81	

Fuente: Instituto Nacional de Estadística (INE).

La serie de suspensiones de pagos ofrece información trimestral desde 1990. La tabla 8.4 recoge los últimos catorce años completos 1990-2003 y los tres primeros trimestres de 2004. La serie alcanza su máximo histórico en el primer trimestre de 1993 con 397 expedientes de suspensión de pagos, a partir de entonces empieza a descender hasta que en 1998 alcanza un valor relativamente estable, alrededor del cual sólo se producen pequeñas oscilaciones. Esta evolución coincide con la recuperación económica producida en España en ese período.

En el siguiente gráfico se puede observar la evolución de la serie con periodicidad trimestral a lo largo de ese período, lo que facilita la interpretación anterior. Se corrobora que tras un fuerte ascenso desde 1990 a 1993, se produce un pronunciado descenso desde 1993 hasta finales de 1997, alcanzándose una relativa estabilidad a partir de 1998.

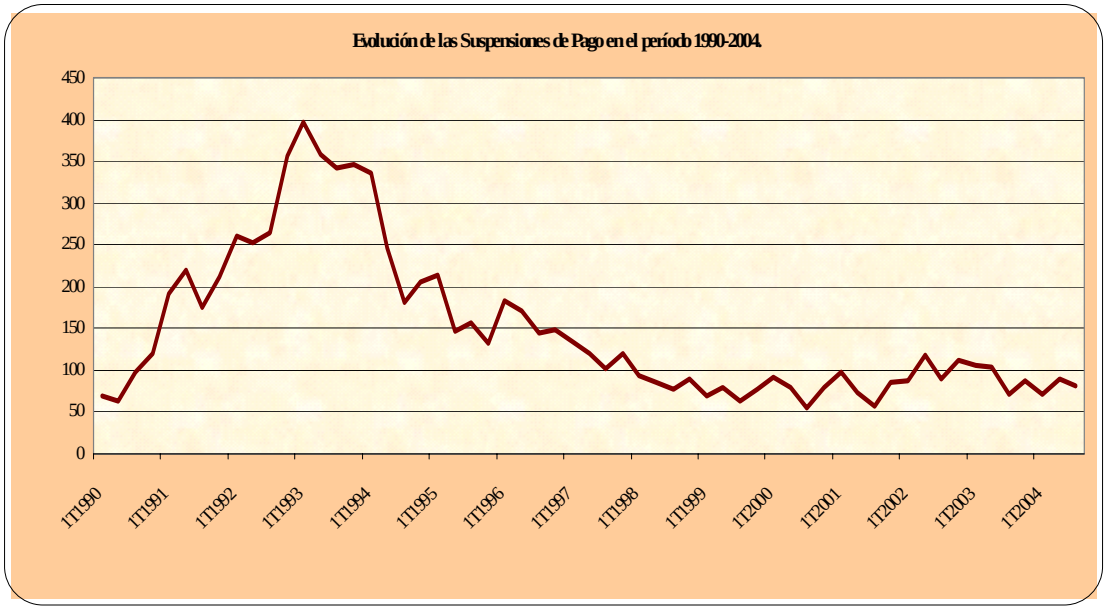


Figura 8.1 Evolución de las Suspensiones de Pagos en España.

A continuación se estudia la distribución de las suspensiones de pagos por tipo de empresa, por causa de la suspensión y, finalmente, por el sector de actividad económica al que pertenece la empresa.

TABLA 8.5 SUSPENSIONES DE PAGOS POR TIPO DE EMPRESA.									
		Persona física		SA		SL		Otras	
	Total	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%
1990	351	46	13,1	242	68,9	49	14,0	14	4,0
1991	798	89	11,2	564	70,7	119	14,9	26	3,3
1992	1.135	139	12,2	713	62,8	236	20,8	47	4,1
1993	1.446	127	8,8	886	61,3	388	26,8	45	3,1
1994	969	73	7,5	590	60,9	274	28,3	32	3,3
1995	650	38	5,8	405	62,3	193	29,7	14	2,2
1996	649	35	5,4	367	56,5	229	35,3	18	2,8
1997	479	23	4,8	255	53,2	188	39,2	13	2,7
1998	348	11	3,2	157	45,1	156	44,8	24	6,9
1999	290	12	4,1	128	44,1	144	49,7	6	2,1
2000	306	11	3,6	131	42,8	154	50,3	10	3,3
2001	314	4	1,3	129	41,1	171	54,5	10	3,2
2002	408	9	2,2	162	39,7	226	55,4	11	2,7
2003	367	9	2,5	140	38,1	205	55,9	13	3,5
2004	324	6	1,9	104	32,1	197	60,8	17	5,2

Fuente: Elaboración propia a partir de INE.

Lo más destacado de la evolución de las suspensiones de pagos por tipo de empresa en el período 1990-2004, es el intercambio de importancia que se ha producido entre las sociedades anónimas (SA) y las sociedades limitadas (SL). Las sociedades anónimas han pasado de suponer un 68,9% del total de suspensiones en 1990 a representar un 31,8% sobre el total en 2004. Por el contrario, las sociedades limitadas han incrementado su presencia desde el 14% inicial hasta un 61,6% en el último año. También se ha reducido notablemente el porcentaje de personas físicas que se declaran en suspensión de pagos.

TABLA 8.6 SUSPENSIONES DE PAGOS POR CAUSA DE SUSPENSIÓN.									
		Falta de liquidez		Escasa demanda		Baja productividad		Otras	
	Total	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%
1990	351	233	66,4	57	16,2	5	1,4	56	16,0
1991	798	475	59,5	150	18,8	17	2,1	156	19,5
1992	1.135	661	58,2	226	19,9	22	1,9	226	19,9
1993	1.446	784	54,2	350	24,2	23	1,6	289	20,0
1994	969	464	47,9	209	21,6	18	1,9	278	28,7
1995	650	301	46,3	135	20,8	12	1,8	202	31,1
1996	649	333	51,3	138	21,3	9	1,4	169	26,0
1997	479	238	49,7	88	18,4	9	1,9	144	30,1
1998	348	175	50,3	45	12,9	11	3,2	117	33,6
1999	290	158	54,5	34	11,7	5	1,7	93	32,1
2000	306	164	53,6	40	13,1	11	3,6	91	29,7
2001	314	173	55,1	44	14,0	7	2,2	90	28,7
2002	408	230	56,4	64	15,7	11	2,7	103	25,2
2003	367	196	53,4	57	15,5	5	1,4	109	29,7
2004	324	193	59,4	51	15,7	4	1,2	77	23,7

Fuente: Elaboración propia a partir de INE.

Si se analiza la evolución de las suspensiones de pagos en función de cuál es la causa que motiva la misma, la causa más importante es la falta de liquidez de la organización suponiendo alrededor de un 60% en el último año. Aunque la evolución del peso de las suspensiones por falta de liquidez ha seguido una forma de U en este periodo, desde un 66,4% inicial disminuye hasta tocar fondo en 1995 y a partir de 1998, empieza un suave ascenso hasta el 59,4% de 2004. La escasez de demanda cobró especial relevancia en los años de crisis de 1993-1996, situándose por encima del 20%, en la actualidad supone algo menos del 16%. Por último, la baja productividad únicamente de declara como causa de la suspensión de pagos en un 1,2% en 2004, tomando su valor más alto en este período en el año 2000 con un 3,6%.

En lo relativo a la distribución de suspensiones de pagos por sectores de actividad, como puede verse en la tabla 8.7, los más importantes son la industria y los servicios, en tercer lugar la construcción, y por último, la agricultura.

TABLA 8.7 SUSPENSIONES DE PAGOS POR SECTOR DE ACTIVIDAD									
		Agricultura		Industria		Construcción		Servicios	
	Total	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%
1993	1.446	34	2,4	609	42,1	167	11,5	636	44,0
1994	969	21	2,2	425	43,9	135	13,9	388	40,0
1995	649	19	2,9	274	42,2	109	16,8	247	38,1
1996	649	11	1,7	272	41,9	126	19,4	240	37,0
1997	479	9	1,9	181	37,8	90	18,8	199	41,5
1998	348	10	2,9	149	42,8	64	18,4	125	35,9
1999	290	6	2,1	120	41,4	51	17,6	113	39,0
2000	306	7	2,3	120	39,2	57	18,6	122	39,9
2001	314	7	2,2	123	39,2	48	15,3	136	43,3
2002	408	13	3,2	165	40,4	51	12,5	179	43,9
2003	367	5	1,4	153	41,7	49	13,4	160	43,6
2004	324	12	3,6	165	50,8	45	13,9	103	31,7

Fuente: Elaboración propia a partir de INE.

La tabla 8.8 recoge la evolución de los activos y los pasivos, en miles de euros, de las empresas que suspendieron pagos en el período 1990-2004, ambas series han evolucionado bastante en paralelo y, también, de manera coherente al número de empresas que presentaban suspensión de pagos. Esto se refleja en las series de activos y pasivos medios que, salvo en 1992 donde marcan su máximo valor, oscilan alrededor de sus valores medios para este período 4.747 y 3.621, respectivamente para el activo medio y el pasivo medio.

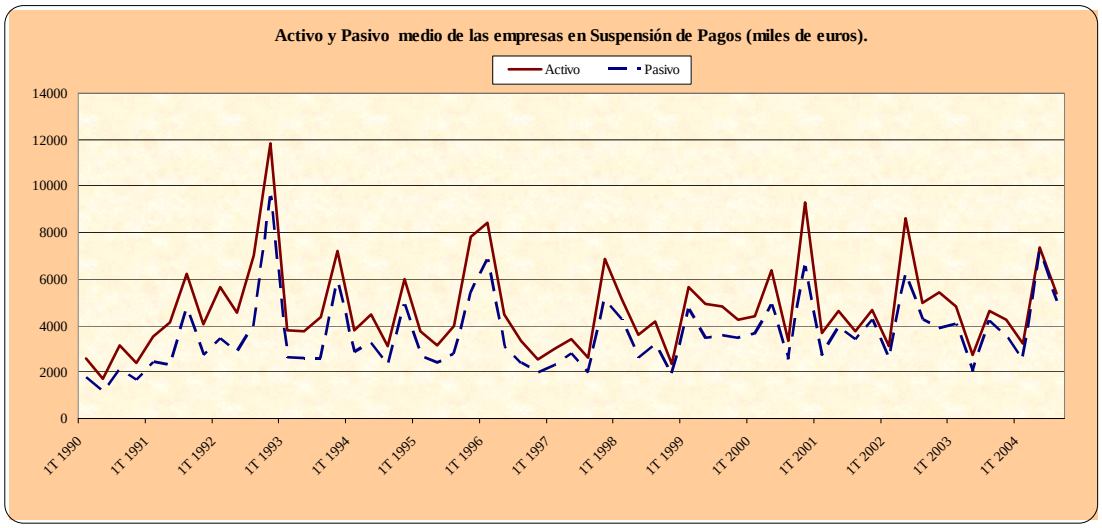


Figura 8.2 Evolución de los activos y pasivos medios de las empresas en suspensión de pagos.

TABLA 8.8 EVOLUCIÓN DE ACTIVOS Y PASIVOS DE LAS EMPRESAS EN S. PAGOS (miles de euros)					
	Nº Susp	Activo	Pasivo	A. Medio	P. Medio
1990	351	885.439	606.097	2.523	1.727
1991	798	3.532.131	2.372.100	4.426	2.973
1992	1.135	8.703.991	6.098.194	7.669	5.373
1993	1.446	6.854.929	4.901.940	4.741	3.390
1994	969	4.181.673	3.182.720	4.315	3.285
1995	650	2.934.922	2.076.592	4.515	3.195
1996	649	3.186.124	2.421.803	4.909	3.732
1997	479	1.912.288	1.471.584	3.992	3.072
1998	348	1.333.345	1.043.046	3.831	2.997
1999	290	1.422.925	1.094.911	4.907	3.776
2000	306	1.833.605	1.382.021	5.992	4.516
2001	314	1.314.142	1.116.193	4.185	3.555
2002	408	2.349.382	1.779.808	5.758	4.362
2003	367	1.489.440	1.244.600	4.058	3.391
2004	324	1.746.159	1.613.414	5.385	4.976

Fuente: Elaboración propia a partir de INE.

8.4.2. Quiebras

La serie de quiebras ofrece información trimestral desde 1990. La tabla 8.9 recoge los últimos catorce años completos 1990-2003 y los tres primeros trimestres de 2004. En la evolución de la serie pueden distinguirse tres periodos, según su tendencia al alza o a la baja. El primero de ellos con tendencia alzista llegaría hasta los años 1995-1996, donde toca techo en el segundo trimestre de 1996 con 199 empresas quebradas. A partir de ahí empezaría a bajar hasta tocar fondo en el tercer trimestre de 2001, con 80 empresas quebradas. Éste es de nuevo un punto de inflexión, a partir del cuál comienza una nueva escalada hasta el final, siendo el último valor disponible para el tercer trimestre de 2004 el máximo de la serie.

TABLA 8.9 EMPRESAS EN QUIEBRA EN ESPAÑA.				
	1 TRIM.	**2 TRIM.**	**3 TRIM.**	**4TRIM.**
1990	27	35	29	41
1991	76	88	74	114
1992	134	122	119	132
1993	159	150	117	192
1994	189	195	141	172
1995	197	192	127	179
1996	192	199	139	183
1997	175	174	115	192
1998	122	154	127	145
1999	124	126	105	129
2000	144	141	104	133
2001	132	116	80	117
2002	143	157	142	187
2003	178	170	131	166
2004	150	135	216	

Fuente: INE.

Para corroborar lo apuntado anteriormente de manera gráfica, puede observarse la figura 8.3.

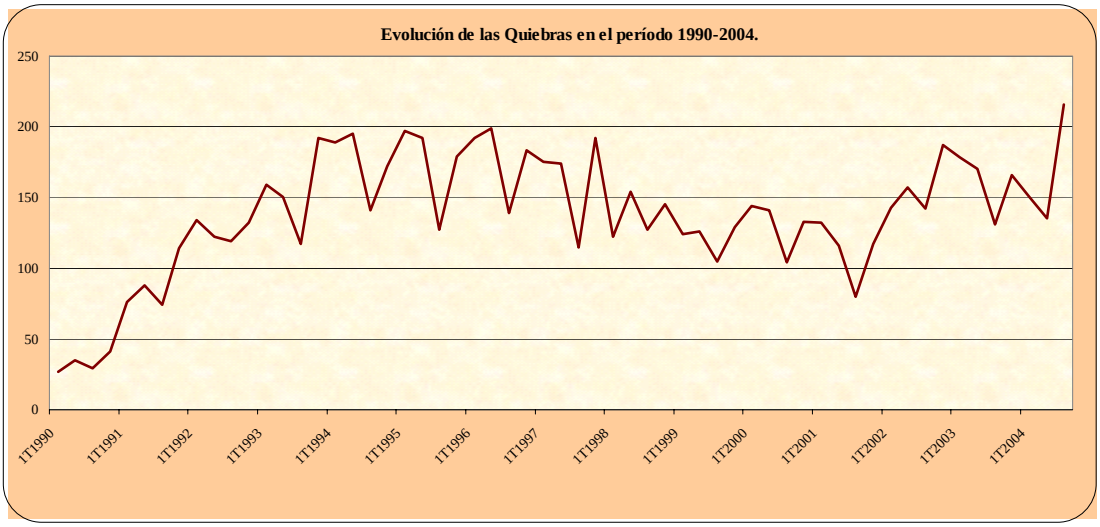


Figura 8.3 Evolución de las Quiebras en España

A continuación se estudia la distribución de las quiebras por tipo de empresa, por causa de la suspensión y, finalmente, por el sector de actividad económica al que pertenece la empresa.

TABLA 8.10 QUIEBRAS POR TIPO DE EMPRESA.

	Total	Persona física		SA		SL		Otras	
		V. Absoluto	%	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%
1990	132	14	10,6	99	75,0	17	12,9	2	1,5
1991	352	28	8,0	264	75,0	47	13,4	13	3,7
1992	507	37	7,3	388	76,5	72	14,2	10	2,0
1993	618	37	6,0	418	67,6	144	23,3	19	3,1
1994	697	30	4,3	451	64,7	198	28,4	18	2,6
1995	695	26	3,7	405	58,3	239	34,4	25	3,6
1996	713	26	3,6	388	54,4	282	39,6	17	2,4
1997	656	20	3,0	312	47,6	302	46,0	22	3,4
1998	548	13	2,4	216	39,4	305	55,7	14	2,6
1999	484	16	3,3	192	39,7	265	54,8	11	2,3
2000	522	13	2,5	178	34,1	310	59,4	21	4,0
2001	445	8	1,8	147	33,0	277	62,2	13	2,9
2002	629	7	1,1	162	25,8	444	70,6	16	2,5
2003	645	7	1,1	167	25,9	452	70,1	19	2,9
2004	668	3	0,4	159	23,8	491	73,5	16	2,3

Fuente: Elaboración propia a partir del INE.

Lo más destacado de la evolución de las quiebras por tipo de empresa en el período 1990-2004, es el intercambio de importancia que se ha producido entre las sociedades anónimas (SA) y las sociedades limitadas (SL). Las sociedades anónimas han pasado de suponer un 75,0% del total de quiebras en 1990 a representar un 23,8% sobre el total en 2004. Por el contrario, las sociedades limitadas han incrementado su presencia desde el 12,9% inicial hasta un 73,5% en el último año. También se ha reducido notablemente el porcentaje de personas físicas que se declaran en quiebra.

Si se analiza la evolución de las quiebras en función de cuál es la causa que motiva la misma, la causa más importante es la insolvencia fortuita, aunque ha visto disminuir su importancia del 39,8% de 1993 al 10,4% en 2004. Seguida de la suspensión de pagos que empieza siendo la causa del 13,1% de las quiebras y acaba representando

únicamente el 5% de las mismas. En la mayoría de las quiebras la causa queda sin determinar.

TABLA 8.11 QUIEBRAS SEGÚN SU CLASE							
	Total	S.Pagos	Insolv. Fort.	Insolv. Culp	Insolv. Fraud.	Alzamiento	Sin det
1993	618	81	246	21	13	14	243
1994	697	129	185	15	10	17	341
1995	695	115	129	7	8	5	431
1996	713	91	103	6	5	2	506
1997	656	57	84	4	5	5	501
1998	548	46	53	2	5	3	440
1999	484	40	53	2	7	1	381
2000	522	41	58	3	3	1	416
2001	445	35	54	3	2	1	349
2002	629	38	72	1	3	0	515
2003	645	35	65	2	2	1	540
2004	668	34	70	0	0	1	563

Fuente: INE.

TABLA 8.12 QUIEBRAS SEGÚN SU CLASE EN PORCENTAJE							
	Total	S.Pagos	Insolv. Fort.	Insolv. Culp	Insolv. Fraud.	Alzamiento	Sin det
1993	618	13,1	39,8	3,4	2,1	2,3	39,3
1994	697	18,5	26,5	2,2	1,4	2,4	48,9
1995	695	16,5	18,6	1,0	1,2	0,7	62,0
1996	713	12,8	14,4	0,8	0,7	0,3	71,0
1997	656	8,7	12,8	0,6	0,8	0,8	76,4
1998	548	8,4	9,7	0,4	0,9	0,5	80,3
1999	484	8,3	11,0	0,4	1,4	0,2	78,7
2000	522	7,9	11,1	0,6	0,6	0,2	79,7
2001	445	7,9	12,1	0,7	0,4	0,2	78,4
2002	629	6,0	11,4	0,2	0,5	0,0	81,9
2003	645	5,4	10,1	0,3	0,3	0,2	83,7
2004	668	5,0	10,4	0,0	0,0	0,2	84,3

Fuente: Elaboración propia a partir del INE.

En lo relativo a la distribución de quiebras por sectores de actividad, como puede verse en la tabla 8.13, el más importante es el sector servicios (46% de media), seguido de la industria (33% de media), en tercer lugar la construcción (19% de media) y por último, la agricultura (1,5% en media).

TABLA 8.13 QUIEBRAS POR SECTOR DE ACTIVIDAD									
		Agricultura		Industria		Construcción		Servicios	
	Total	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%	V. Absoluto	%
1993	618	15	2,4	205	33,2	98	15,9	300	48,5
1994	697	8	1,1	252	36,2	131	18,8	306	43,9
1995	695	6	0,9	243	35,0	122	17,6	324	46,6
1996	713	8	1,1	263	36,9	130	18,2	312	43,8
1997	656	10	1,5	235	35,8	116	17,7	295	45,0
1998	548	2	0,4	186	33,9	124	22,6	236	43,1
1999	484	11	2,3	165	34,1	95	19,6	213	44,0
2000	522	11	2,1	159	30,5	110	21,1	242	46,4
2001	445	8	1,8	134	30,1	82	18,4	221	49,7
2002	629	10	1,6	178	28,3	131	20,8	310	49,3
2003	643	10	1,6	218	33,9	116	18,0	299	46,5
2004	668	11	1,6	213	31,9	133	19,9	311	46,6

Fuente: Elaboración propia a partir del INE.

TABLA 8.14 EVOLUCIÓN DE ACTIVOS Y PASIVOS DE LAS EMPRESAS EN QUIEBRA (miles de euros)					
	Nº Quiebras	Activo	Pasivo	A. Medio	P. Medio
1990	132	63.314	140.319	480	1.063
1991	352	192.349	288.059	546	818
1992	507	351.401	500.262	693	987
1993	618	413.141	606.956	669	982
1994	697	545.604	777.163	783	1.115
1995	695	659.017	1.008.390	948	1.451
1996	713	406.632	612.323	570	859
1997	656	324.144	568.295	494	866
1998	548	276.654	517.832	505	945
1999	484	389.649	623.004	805	1.287
2000	522	312.856	702.824	599	1.346
2001	445	240.369	481.391	540	1.082
2002	629	1.037.848	1.745.385	1.650	2.775
2003	645	525.351	749.138	814	1.161
2004	668	454.229	744.364	680	1.115

Fuente: Elaboración propia a partir del INE

La tabla 8.14 recoge la evolución de los activos y los pasivos, en miles de euros, de las empresas que quebraron en el período 1990-2004, ambas series han evolucionado bastante en paralelo y, también, de manera coherente al número de empresas quebradas.

Esto se refleja en las series de activos y pasivos medios que, salvo en 2002 donde marcan su máximo valor, oscilan alrededor de sus valores medios para este período 718,49 y 1.190,17 respectivamente para el activo medio y el pasivo medio por empresa (en miles de euros).

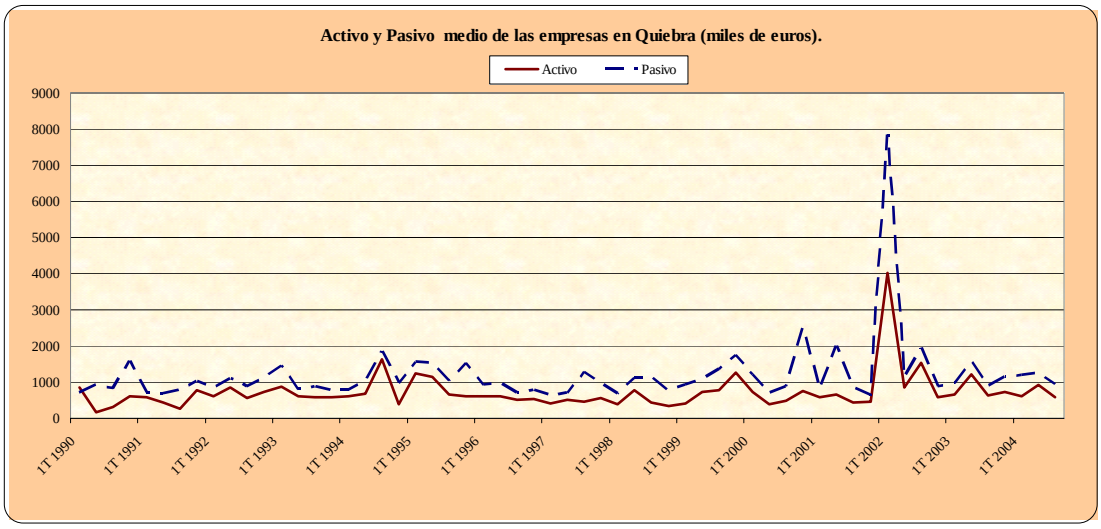


Figura 8.4 Activos y Pasivos medios de las empresas en quiebra.

8.4.3. Sociedades mercantiles disueltas y fusionadas

TABLA 8.15 SOCIEDADES MERCANTILES DISUELTAS EN ESPAÑA.

	1 TRIM.	2 TRIM.	3 TRIM.	4TRIM.
1993	2.412	2.106	1.196	1.253
1994	1.416	1.039	676	892
1995	1.207	941	752	987
1996	5.536	4.079	4.181	3.658
1997	3.149	2.191	1.417	3.094
1998	2.429	1.749	1.495	1.792
1999	2.416	1.576	1.154	2.611
2000	2.394	1.458	1.106	1.751
2001	2.392	1.500	1.178	1.769
2002	2.768	1.787	1.282	1.951
2003	2.483	1.753	1.589	2.423
2004	4.552	2.421	1.736	

Fuente: INE

La estadística sobre sociedades mercantiles disueltas tiene periodicidad mensual, aunque se presenta aquí de manera trimestral para facilitar la comparación con las series de suspensiones de pagos y quiebras. Esta serie, como muestra la figura 8.5, presenta una forma de hoja de sierra característica de las series con un marcado carácter estacional, pero su tendencia es sólo ligeramente ascendente.

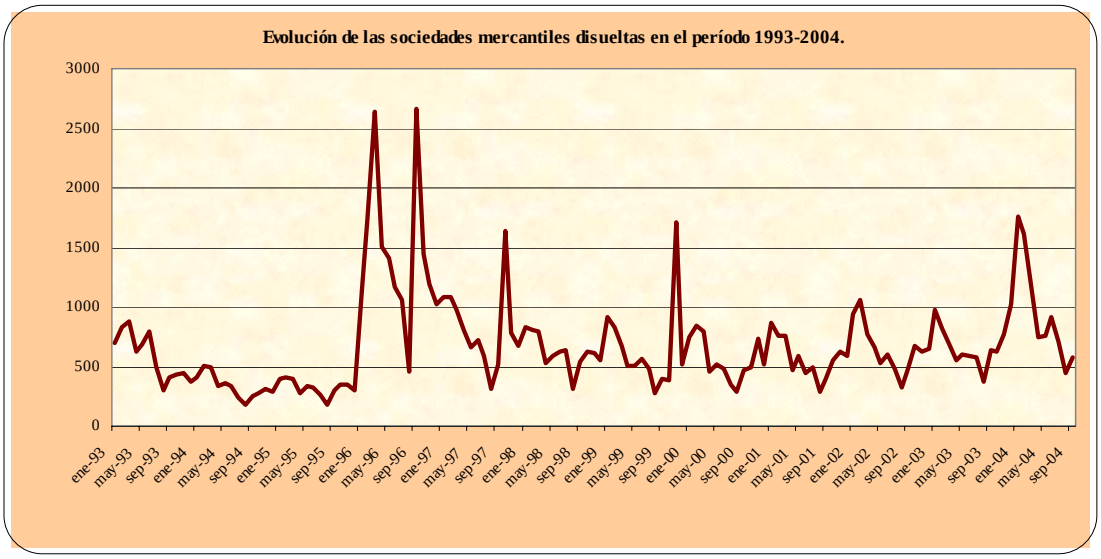


Figura 8.5 Evolución de las empresas disueltas en España.

Por lo que respecta a las sociedades mercantiles fusionadas, su evolución se recoge en la tabla 8.16. Empieza este período con un ligero descenso hasta 1995, a partir de ese año comienza una tendencia ascendente que llega hasta los años 2001-2002. Desde entonces la serie emprende un ligero descenso hasta los valores actuales.

TABLA 8.16 SOCIEDADES MERCANTILES FUSIONADAS EN ESPAÑA.				
	1 TRIM.	**2 TRIM.**	**3 TRIM.**	**4TRIM.**
1993	276	194	148	173
1994	218	113	93	189
1995	163	72	105	113
1996	175	77	123	257
1997	186	123	102	264
1998	257	116	165	343
1999	278	173	220	405
2000	395	281	251	461
2001	459	223	240	525
2002	424	262	283	501
2003	333	239	284	430
2004	328	223	296	

Fuente: INE.

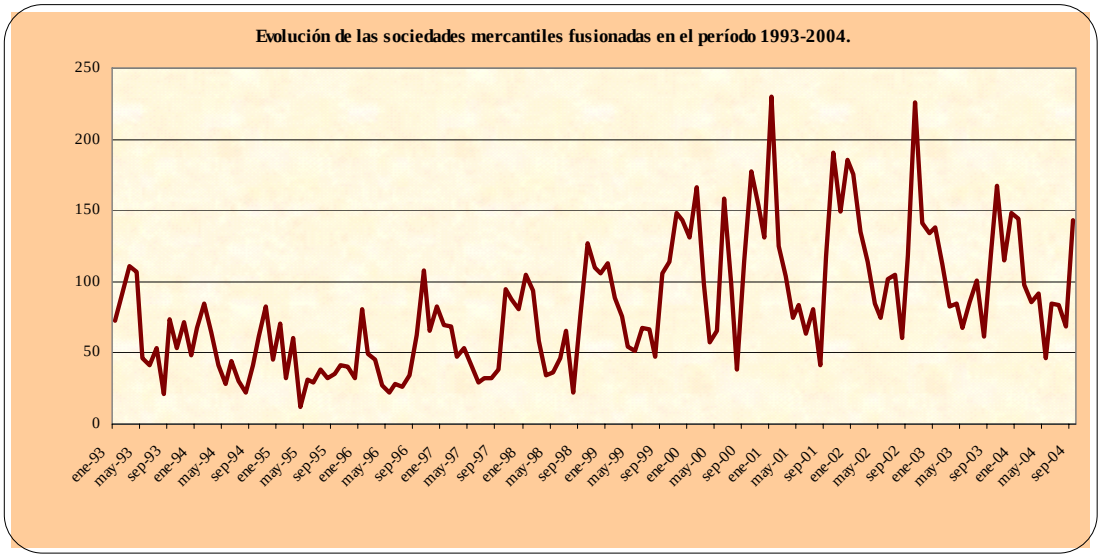


Figura 8.6 Evolución de las empresas fusionadas en España.

8.4.4. Evolución comparada de los distintos tipos de fracaso

Para poder establecer mejor las comparaciones entre la evolución que han seguido los cuatro tipo de empresas consideradas (suspensiones de pagos, quiebras, disueltas y fusionadas) se puede eliminar el efecto de la magnitud de cada una de ellas, para ello se calculan los índices considerando como referencia el promedio de cada serie en el año 1993 (base 1993=100). Así, en la figura 8.7 se puede ver como las series que

mayores fluctuaciones han sufrido en términos relativos han sido las de “disueltas” y “fusiones”. Oscilando la serie de “quiebras” alrededor de su valor inicial y, finalmente, la serie de “suspensiones de pagos ” reduciéndose hasta una quinta parte de su nivel inicial.

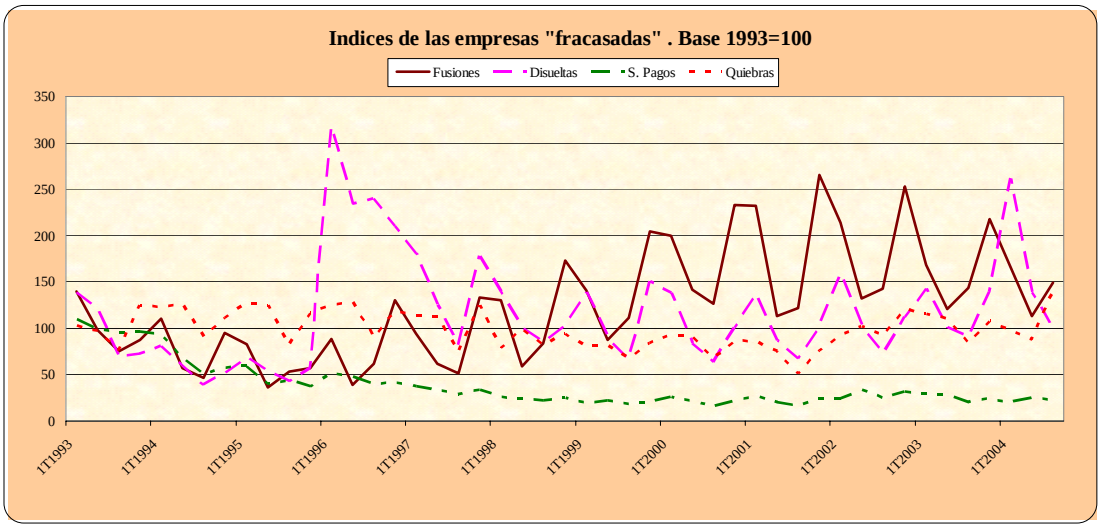


Figura 8.7 Indices de las empresas fracasadas en España.

Para acabar este análisis descriptivo, se ofrecen las series desestacionalizadas y las series de ciclo tendencia, para los cuatro tipos de empresas mencionados.

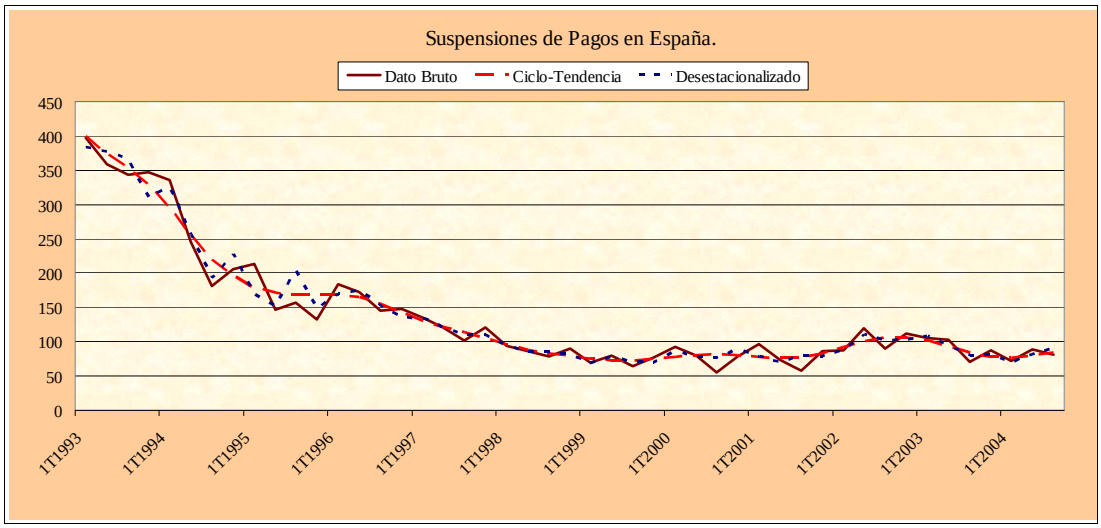


Figura 8.8 Suspensiones de pagos en España.

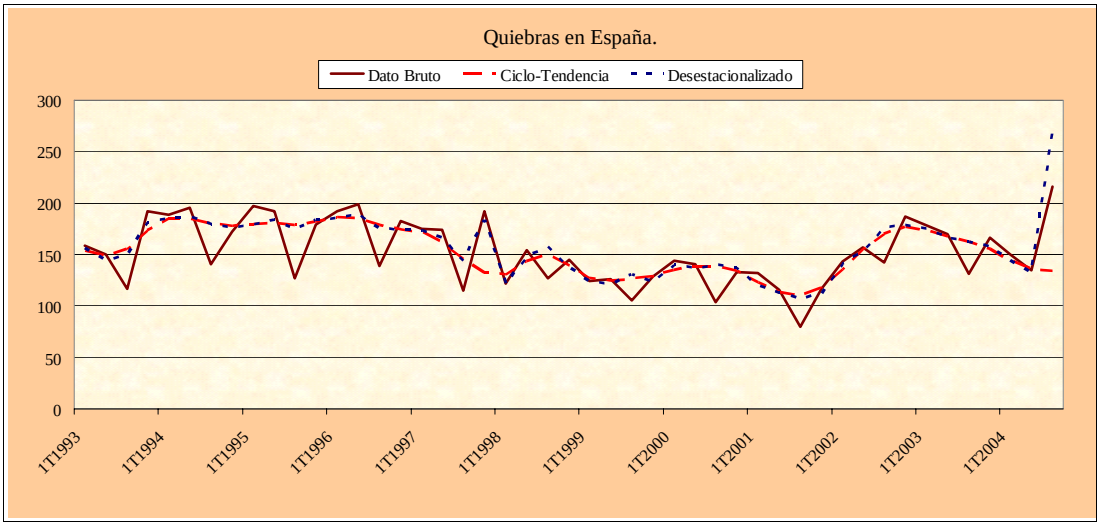


Figura 8.9 Quiebras en España.

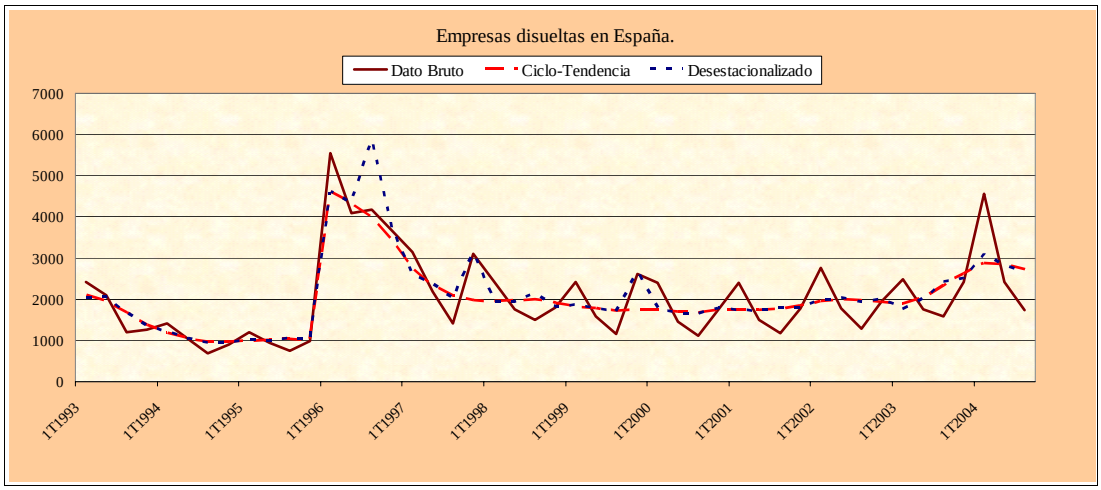


Figura 8.10 Empresas disueltas en España.

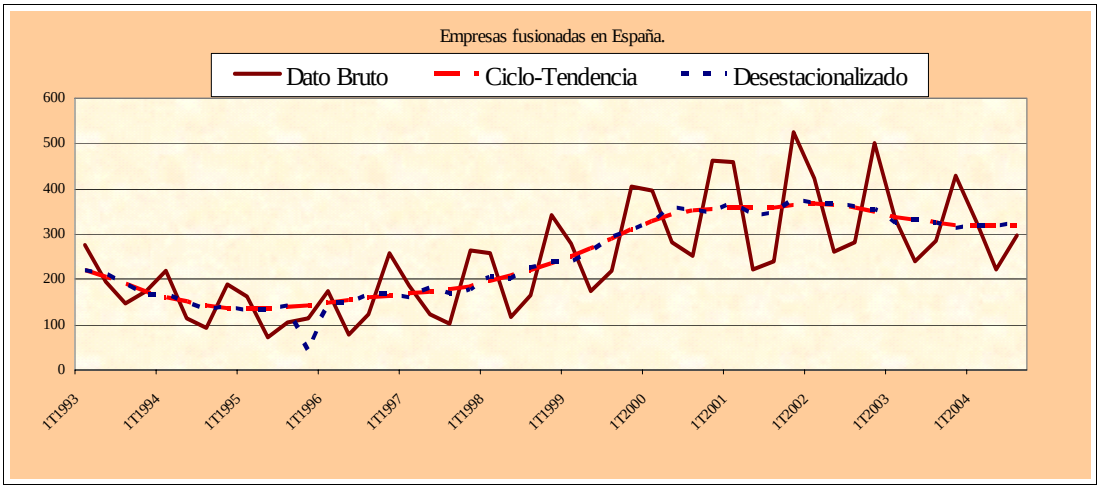


Figura 8.11 Empresas fusionadas en España.

Una vez puesto de manifiesto la relevancia de la predicción del fracaso empresarial, analizado el estado actual del pronóstico del fallo empresarial y realizado el análisis descriptivo de las suspensiones de pagos, quiebras, fusiones y disoluciones en España durante los últimos años, se abordará en el capítulo siguiente la tarea de predecir el fracaso de una empresa de manera anticipada con información de uno o varios años antes de que se produzca el fracaso. En primer lugar, se diferenciará entre empresas activas y fracasadas, abordando posteriormente la tarea más compleja de discriminar entre más de un grupo de fracaso.

CAPÍTULO 9

PREDICCIÓN DEL FRACASO EMPRESARIAL MEDIANTE EL MÉTODO BOOSTING

9.1. INTRODUCCIÓN

Una vez expuestos los principales modelos de combinación de clasificadores, en este capítulo se va a realizar un estudio sobre la predicción del fracaso empresarial para un conjunto de empresas españolas.

Ya en el capítulo anterior se ha puesto de manifiesto el interés de esta investigación empírica, destacando fundamentalmente las repercusiones del fracaso de las empresas en el entorno económico y social de las mismas, y lo novedoso de la técnica que se va a utilizar. La predicción del fracaso empresarial es un campo de investigación ampliamente consolidado, ya que se inició a finales de la década de los sesenta. Si bien, ha sido en la última década, a lo largo de los noventa, cuando se han ido superando algunas limitaciones en el mismo, como puede ser la consideración de modelos no paramétricos (redes neuronales y árboles de clasificación, fundamentalmente), que no

se ven sujetos a supuestos tan restrictivos, tales como la normalidad de las variables utilizadas o la forma funcional que las liga.

Además, en la mayoría de los trabajos realizados en la predicción del fracaso empresarial se trabaja considerando únicamente dos grupos de empresas "fracasadas" y "sanas". En España en el grupo de empresas fracasadas se suelen incluir las empresas en quiebra o suspensión de pagos. Pues bien, en este trabajo en primer lugar se va a utilizar el esquema dicotómico habitual, discriminando entre empresas sanas y empresas fracasadas. Si bien, dentro de las empresas fracasadas se incluyen, además de las suspensiones de pagos y las quiebras, las empresas que han sido absorbidas o disueltas, considerando de esta manera el fracaso desde una perspectiva amplia, que tiene en cuenta tanto el fracaso legal como el fracaso como cese, ya sea por cambio en la propiedad o por fin de la actividad. Ahora bien, se ha adoptado esta definición a sabiendas del riesgo que se corre, ya que no existe un consenso generalizado en este sentido en la literatura sobre fracaso empresarial, y por tanto, esta definición puede ser tan válida como otras muchas.

Una vez realizada la predicción para el caso dicotómico, las empresas fracasadas se dividirán en dos grupos, por un lado las empresas absorbidas o disueltas, que se incluirán dentro del fracaso1 y, por otro, las empresas en suspensión de pagos o quiebra, que estarán dentro del fracaso2. De esta forma, se podrá constatar si las empresas de fracaso1, son un grupo claramente diferenciado de las activas. Así como, si se puede discriminar, en base a la información disponible, entre las empresas pertenecientes a fracaso1 y fracaso2.

Este capítulo comienza con una breve visión teórica de cómo debe tratarse la información antes de su uso en cualquier estudio estadístico en general, y en clasificación en particular. A continuación, se describe el proceso de muestreo y depuración de los datos, para pasar en tercer lugar a la aplicación de diversos modelos con objeto de predecir el fracaso empresarial, estableciendo finalmente los principales resultados.

9.2. ALGUNAS CONSIDERACIONES TEÓRICAS SOBRE EL TRATAMIENTO DE LA INFORMACIÓN

9.2.1. Introducción

En clasificación, como en cualquier otro campo de la Estadística Multivariante, se pueden considerar cinco fases fundamentales en la realización de una aplicación. A continuación se recogen estas cinco fases:

1.Definición del problema de investigación.
   a. Identificación de objetivos e hipótesis.
   b. Desarrollo del modelo.
   c. Selección de variables y su medida.
   d. Elección de la técnica o técnicas que se van a utilizar.
2.Diseño del análisis.
   a. Implementación del modelo en la técnica seleccionada.
   b. Tamaño muestral.
   c. Recogida de datos.
3.Análisis previo de los datos.
   a. Identificación y tratamiento de outliers o casos atípicos.
   b. Identificación y tratamiento de datos ausentes.
   c. Validación de los supuestos de la técnica seleccionada.
4.Estimación y evaluación del modelo.
5.Interpretación de resultados.

A pesar de la ordenación teórica que, para mayor claridad en la exposición, se ha realizado, hay que destacar que en la práctica las etapas señaladas anteriormente no están perfectamente delimitadas en el tiempo, sino que, en muchos casos, se solapan unas con otras. Se aborda, por tanto, en esta sección la tarea concerniente a la selección de las variables de entrada, análisis previo de los datos, y cuando sea necesario la transformación de los mismos antes de su presentación al clasificador boosting.

9.2.2. Obtención del conjunto final de variables de entrada

Para comenzar, se trata la tarea de selección de los inputs o variables de entrada. Esta primera fase es fundamental si se quiere evitar la pérdida de tiempo realizando aplicaciones a ciegas que darían lugar a resultados irrelevantes. Habitualmente, cuando se afronta un problema de regresión o uno de clasificación, se cuenta con un numeroso conjunto de posibles variables explicativas a utilizar. Sin embargo, en general no es conveniente incluir todas aquellas variables en el análisis sin más, debido a que algunas de ellas pueden resultar poco útiles para el propósito en cuestión. Además, el problema no es sólo que haya variables que no aporten información relevante para la reducción del error cometido sino que, debido al problema conocido como "maldición de la dimensión" y dado que el conjunto de datos de entrenamiento suele ser bastante limitado, es muy probable que se incurra en una situación de sobre-entrenamiento del clasificador, con la consecuente falta de credibilidad sobre las estimaciones de los parámetros desconocidos. Relacionado con este problema está el conocido intercambio entre el sesgo y la varianza, componentes, como se ha visto en este trabajo, del error de generalización. Muy pocas variables de entrada reducen el espacio de búsqueda, dando lugar a un posible incremento del sesgo. Sin embargo, demasiadas variables hacen que el clasificador se vuelva muy complejo, aumentando la contribución de la parte de la varianza al error de generalización.

En este punto, resulta conveniente puntualizar la distinción entre dos tareas que, aunque muy relacionadas e incluso a veces confundidas, responden a criterios distintos y, por tanto, se realizan también mediante técnicas distintas. Estas tareas son: selección y extracción de rasgos. En primer lugar, el término *selección de rasgos* se refiere a algoritmos que seleccionan el "mejor" subconjunto de variables de entrada a partir del conjunto original de variables. Los árboles de clasificación realizan esta labor de forma automática. Por otra parte, el término *extracción de rasgos* se refiere a la creación de un conjunto de variables mediante la transformación o combinación de las variables originales. Los ratios utilizados en la aplicación se obtienen combinando dos o más variables contables.

9.2.3. Análisis previo de los datos

Seleccionado, al menos en principio, el conjunto de variables que se van a utilizar en la aplicación, es conveniente realizar un análisis previo de los datos para lograr un conocimiento básico de los mismos, estudiando con sencillas técnicas estadísticas y representaciones gráficas la distribución de cada una de las variables de manera aislada, así como las relaciones entre variables.

9.2.3.1. - Detección de valores atípicos

Un aspecto importante a tratar en el primer contacto con los datos es el relacionado con la identificación de **casos atípicos**, es decir, observaciones que caen fuera de los rangos de la distribución una vez establecidos los valores umbrales. Este paso es fundamental en cualquier técnica, pero especialmente en clasificación, la existencia de valores extremos hace que al normalizar o estandarizar las variables (práctica frecuente como se estudiará a continuación), las nuevas variables tendrán un rango de variación efectivo (exceptuando los extremos) excesivamente pequeño.

La detección de estas observaciones se puede llevar a cabo mediante la estandarización de variables desde una perspectiva univariante, considerando pares de variables y realizando gráficos de dispersión e incluso tratar el aspecto desde una perspectiva multivariante utilizando, por ejemplo, la Distancia de Mahalanobis (D) o su cuadrado (D^2):

$$D^2 = \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)' \boldsymbol{S}^{-1} \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)$$

Se trata de una medida de distancia de cada observación en un espacio multidimensional respecto al centro medio de las observaciones que, además, tiene en cuenta la correlación entre las variables.

Una vez detectadas las observaciones atípicas hay que decidir si se eliminan de la muestra, evitando así la distorsión que estos datos pueden provocar en el análisis, o bien, optar por mantenerlas para no perder la generalidad de los resultados.

9.2.3.2. - Detección de existencia de ruido

Otro problema frecuente que aparece en la investigación empírica y, especialmente en datos económicos y financieros, es la presencia de ruido. Este hecho degenera la capacidad de aprendizaje y generalización del clasificador. Los ejemplos maliciosos son aquéllos que se encuentran muy cerca de la frontera real que separa unas clases de otras (en un problema de clasificación). La figura 9.1 muestra el problema gráficamente. En ella se reproduce un problema de clasificación con dos variables de entrada (X e Y) y dos clases (A y B). M representa un conjunto de patrones que aunque presentan pares de valores (*x*,*y*) muy similares, pertenecen a clases distintas.

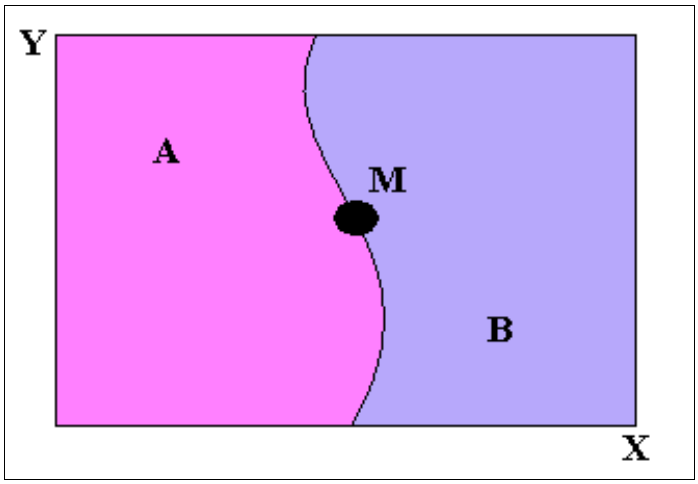


Figura 9.1: Representación gráfica del problema de ruido catastrófico.

Una posible solución es eliminar esos patrones, lo que haría que la velocidad fuese un poco mayor aunque no mejoraría la capacidad de generalización. Cuando sea posible, se hace recomendable aumentar el número de datos disponible con nuevas

observaciones del área donde se encontraban los maliciosos, aunque esto en la práctica no suele ser posible.

Formalmente, dado un conjunto finito de aprendizaje **T** = (**X**,**Y**) donde **X** f ú^m e Y f ú^n , un procedimiento de aprendizaje heteroasociativo intenta encontrar un mapa tal que f: ú^m 6 ú^n. Cada elemento de **T** tiene la forma ($\boldsymbol{x}_i$, $\boldsymbol{y}_i$) $\boldsymbol{x}_i$ f ú^m, $\boldsymbol{y}_i$ f ú^n . Se define la medida m (**"**,*****) como un grado de la "maliciosidad", de tal forma que dos ejemplos cualesquiera $\boldsymbol{x}_i$ y $\boldsymbol{x}_j$ **,** T son maliciosos con grado de malicia m (**"**,*****) si

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) \# \text{"} \quad \text{y} \quad d(\boldsymbol{y}_i, \boldsymbol{y}_j) > \text{*}$$

siendo *d* una medida de distancia, por ejemplo, la distancia euclídea.

Cuando **"** y ***** son iguales a 0, se está ante el peor caso de maliciosidad (ruido catastrófico). En este caso las variables de entrada son iguales pero la clase es diferente, empeorando la situación cuanto mayor sea la distancia entre las clases (en aquellos problemas en los que se pueda medir).

El incremento del ruido en relación a la información contenida en los datos hace que el resultado, en términos de generalización, sea muy deficiente. Los resultados finales dependerán del modo de aprendizaje.

En general, no es fácil detectar a simple vista cuáles son, si los hay, los patrones de aprendizaje maliciosos y de ellos, cuáles deben ser eliminados de la muestra. Ante un par ($\boldsymbol{x}_i$,$\boldsymbol{x}_j$) de vectores maliciosos, si se quiere eliminar sólo uno de ellos, la selección se puede realizar de manera aleatoria. Otra idea es aumentar la muestra de datos de entrenamiento con el vector $\boldsymbol{x}_i$ o $\boldsymbol{x}_j$ que esté más próximo a la superficie de decisión definida por el clasificador entrenado con la muestra libre de ruido. En las aplicaciones prácticas esto se puede ver como un proceso incremental que requiere varias iteraciones de entrenamiento. En este sentido se pierde rapidez en el entrenamiento pero se gana en capacidad de generalización.

9.2.3.3. - Tratamiento de datos ausentes

Por otra parte, con cierta frecuencia cuando se afronta una investigación empírica se encuentran *datos ausentes* debido fundamentalmente a algunas de las siguientes causas: problemas en el proceso de recolección e introducción de datos, negación a contestar por parte de los encuestados, etc. Los efectos más importantes de la ausencia de datos son la reducción del tamaño de muestra disponible para el análisis y la posibilidad de que esas ausencias escondan sesgos. Se puede solucionar el problema utilizando alguna de las siguientes pautas:

•*Utilizar sólo aquellas observaciones con datos completos.-* Es el método más simple y directo, de ahí que sea la solución propuesta por defecto en la mayor parte de los paquetes estadísticos. Sólo debe utilizarse este método si la ausencia de datos obedece a un proceso aleatorio para evitar obtener resultados sesgados y no generalizables para la población.

•*Sustitución* del caso con todos los datos ausentes, o en gran cantidad, por otro caso, no muestral, que pueda ser similar pero con información completa.

•*Supresión* de las variables que peor se comportan respecto a los datos ausentes.- Es la solución más eficiente en el caso en que la ausencia de datos no sea casual. Debe hacerse una evaluación entre lo que se gana al eliminar una fuente de datos ausentes y la pérdida de información al eliminar variables del análisis.

•*Uso de métodos de imputación.-* Proporcionan una estimación de los valores ausentes basada en valores válidos de la muestra. Se pueden sustituir los datos ausentes de una variable por la media de la misma (moda o probabilidades a priori de cada clase en variables cualitativas), realizar una estimación de los valores mediante un análisis de regresión o una red neuronal utilizando el resto de variables de la muestra, o bien, combinar distintos métodos utilizando finalmente la media de las diversas estimaciones con objeto de evitar los problemas específicos de cada método.

9.2.4. Preprocesamiento de los datos

Entrando, por último, en lo que propiamente se denomina *preprocesamiento* de los datos. Se comienza analizando el proceso de reescalado de los datos, bien mediante *normalización* o bien mediante *estandarización*. Este tipo de preprocesamiento se encuentra relacionado con las unidades de medida de las variables y, por tanto, con el rango de valores que presentan las mismas. Reescalar un conjunto de datos consiste en sumar (o restar) una constante a todas las variables utilizadas, y luego multiplicar (o dividir) por otra constante. Esta operación es útil, por ejemplo, cuando se pretende cambiar las unidades en que viene medida una variable cuantitativa, para conseguir que el rango de ésta sea apropiado para ser introducida en el sistema de clasificación.

Aunque en ocasiones se intercambian los términos normalizar y estandarizar en las operaciones de re-escalado, se podría remarcar sus diferencias mediante las siguientes definiciones. Se entiende por *normalización* el proceso de conversión del rango de una variable cuantitativa a un intervalo de tipo [0,1] o [-1,1]. Por otra parte, se considera como *estandarización* la transformación de una variable cuantitativa en otra con media nula y desviación típica igual a uno.

Bastante más complejo es el procedimiento de estandarización multivariante, mediante el cual se persigue la obtención de una matriz de datos, con vector de medias nulo y matriz de varianzas-covarianzas igual a la matriz identidad, es decir, todas las variables tendrían media nula, desviación típica igual a 1 y, además, estarían incorreladas linealmente. Si en la estandarización univariante, basta con restar a cada valor de cada variable la media de ésta y dividir entre la desviación típica, el proceso multivariante es ligeramente más complicado. A continuación se expone brevemente la forma de llevarlo a cabo. Sea una matriz de datos $\mathbf{X}$, con matriz de varianzas-covarianzas $\mathbf{S_x}$ y sea $\tilde{\mathbf{X}}$, la matriz de datos centrados, la transformación se realiza partiendo de la descomposición espectral de $\mathbf{S_x} = \mathbf{A\,D\,A'}$, quedando:

$$\mathbf{Y} = \tilde{\mathbf{X}}\mathbf{S}_x^{-\frac{1}{2}} = \tilde{\mathbf{X}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{A}'$$

donde, **D** es una matriz diagonal que contiene los autovalores de $\mathbf{S}_x$ y **A** contiene los autovectores. En realidad, este proceso extrae todas las componentes principales normalizadas.

Sobre la conveniencia o no de llevar a cabo la normalización o estandarización de las variables de entrada, hay que analizar para qué tipo de clasificador se van a utilizar los datos. Así, por ejemplo, si las distintas variables de entrada se utilizan como inputs en una red de Kohonen y se van a calcular, por tanto, distancias euclídeas (o alguna otra medida de distancia), entonces grandes diferencias en los rangos de valores de las distintas variables influirán en la importancia relativa de cada una de ellas. En este caso no sólo es recomendable la normalización, sino que se hace imprescindible, a no ser que se esté en la circunstancia de querer potenciar a propósito la importancia de alguna de las variables.

Por otra parte, cuando se trabaja con un árbol de clasificación, parece en principio poco necesario e incluso redundante, reescalar las variables de entrada, pues los árboles no deben verse afectados por esta circunstancia.

Cuando las variables con las que se trabaja son de tipo cualitativo, la transformación previa de los datos se denomina *codificación*. La forma más sencilla que se plantea es sustituir las diferentes modalidades por valores numéricos. Por ejemplo, codificación del sexo de los individuos (*Hombre* = 0 y *Mujer* = 1) o codificación de la forma jurídica que presenta una empresa (*sociedad anónima* = 0, *sociedad limitada* = 1, *otras* = 2). Sin embargo, hay que ser cautos con este tipo de codificación, ya que en gran parte de las ocasiones se estará introduciendo una ordenación arbitraria de las modalidades, que no tienen por qué ser ordinales.

Para salvar este problema se utiliza el método de codificación conocido como *uno-de-N*. Este método consiste en establecer un número de variables numéricas igual al número de modalidades, salvo en el caso de dos modalidades, en el que una variable será suficiente. Así, en el ejemplo de la codificación de la forma jurídica de la empresa, se deberían establecer tres variables de entrada, codificando de la siguiente forma: (*Sociedad anónima* = {1,0,0}, *Sociedad limitada* = {0,1,0}, Otras = {0,0,1}).

A la vista está el problema que plantea este método, y es que en el caso en que se disponga de un número elevado de variables cualitativas de entrada, que a su vez presenten un número elevado de modalidades, el número de variables de entrada se hace demasiado grande, dando lugar al problema de la *maldición de la dimensión*.

9.3. RECOGIDA Y TRATAMIENTO DE LA INFORMACIÓN

Como se ha puesto de manifiesto, el objetivo de este trabajo es la obtención de un sistema automático de clasificación que, una vez entrenado, y a partir de una serie de características de las empresas, sea capaz de predecir si una empresa va a fracasar o no en los próximos ejercicios económicos. De acuerdo a lo establecido en la introducción de este capítulo se distinguirá, en principio, entre empresas activas y fracasadas, para después profundizar dentro de las fracasadas planteando tres clases, activas, fracaso1 y fracaso2. Así, el proceso comienza con la recogida de información necesaria para la consecución del objetivo planteado.

9.3.1. Fuentes de información

La muestra de empresas utilizada para la realización de este trabajo ha sido obtenida de la base de datos SABI (Sistemas de Análisis de Balances Ibérico) de la empresa Informa, S.A., que recoge las cuentas anuales de más de 700.000 empresas españolas y portuguesas[32]. Una vez más, se quiere aprovechar la ocasión para mostrar el

[32] En concreto se ha utilizado la base de datos en CD-Rom SABI Windows, versión 15.0, actualización 56 de abril de 2004, conformada por 645.051 empresas españolas y 78.505 portuguesas

agradecimiento a la empresa Informa, S.A., y en especial en la persona de Marta de la Fuente responsable de la cuenta de la Universidad de Castilla La Mancha, por el apoyo mostrado en los momentos en que ha sido necesario. El periodo de referencia para las empresas fracasadas es el período 2000-2003, como se explica a continuación

Como en todos los estudios sobre fracaso empresarial, a la hora de seleccionar la muestra se centra la atención primero en la clase de empresas menos abundantes, en este caso es la categoría fracaso2, que corresponde a empresas que estuviesen en situación de quiebra o suspensión de pagos, de las que sólo había 590 empresas en el período 2000-2003 que dispusiesen de información completa de todas las variables que se van a utilizar, tanto en el año del fracaso como en los cinco años anteriores. También es normal en este tipo de estudios trabajar con empresas que han fracasado, no en un único año, sino a lo largo de un período de varios años, para poder reunir un número mayor de estas empresas en la muestra. De esta manera la información de las variables debe ser considerada en términos relativos respecto al año del fracaso, que se puede representar por *t*, siendo los años previos *t-1*, *t-2*, *t-3*, *t-4*, *t-5*.

En la clase fracaso1 en el período 2000-2003 se seleccionaron un total de 1.014 empresas que disponían de información completa de todas las variables, tanto para el año del fracaso como para los cinco años previos. Por último, las empresas activas son aquellas que continuaban con su actividad en el año 2003 y que dispusiesen de información de todas las variables para el año 2003 y los cinco ejercicios anteriores, por lo que se recogen un total de 16.768 empresas sanas. De este modo, se parte inicialmente de un conjunto de 18.372 empresas, lo que supone una muestra de tamaño notable si se compara con los tamaños habitualmente utilizados en este tipo de estudios (Laffarga y Mora, 2002).

Finalmente, cuando se dio por terminado el proceso de obtención de información, la base de datos contaba con 18.372 registros (empresas) en los que aparecían los siguientes campos:

<*Estado*

<*Jurídica*

<*CNAE 93*

<*Activo Circulante*

<*Existencias*

<*Tesorería*

<*Activo Total*

<*Fondos Propios*

<*Pasivo Circulante*

<*Pasivo Total*

<*Ventas*

<*Ingresos netos*

<*Beneficio antes de Intereses e Impuestos (BAII)*

<*Cash Flow*

Una vez más se insiste en que para cada variable se recoge información del último año disponible y de los cinco años anteriores, salvo en el caso de las tres primeras variables citadas, que son de carácter cualitativo, y recogen el estado de la empresa (*Estado*), su forma jurídica (*Jurídica*) y el sector al que pertenecen según la *Clasificación Nacional de Actividades Económicas* de 1993 (CNAE93). Por tanto, la base de datos que sirve de base a este trabajo se compone de 69 campos cada uno de ellos con 18.372 registros.

9.3.2. Tratamiento de la información

Una vez recogida la información, el proceso de tratamiento hasta que ésta pudiera ser analizada e incluida en el modelo fue el siguiente. En primer lugar, se calculan los ratios que se van a utilizar en el análisis. Estos ratios son elegidos en base a tres criterios:

1. Haber sido utilizados frecuentemente en la literatura existente de predicción del fracaso empresarial. A falta de una teoría del fracaso empresarial que deje zanjada la

cuestión de qué ratios utilizar, sí que existen algunos ratios comúnmente utilizados que se incluyen también en este trabajo.

2. Disponibilidad de la información. Éste, como otros muchos estudios empíricos, debe ceñirse a la información que existe disponible ante la imposibilidad de realizar una recogida de información propia debido, entre otras cosas, al elevado coste que supondría.

3. Elección propia. En este caso, se ha considerado oportuno incluir el logaritmo del activo circulante, debido a los buenos resultados que ha mostrado en las pruebas realizadas.

Una vez justificada la elección de los ratios, a continuación se aporta una breve descripción de las variables a utilizar, incluyendo los ratios, las variables cualitativas y el logaritmo del Activo Total y del Activo Circulante.

1.- **ESTADO**. Esta variable cualitativa recoge el estado de la empresa, distinguiendo entre fracaso1, activas y fracaso2. Para una mayor claridad en el desarrollo de la explicación, en primer lugar se aborda la tarea más habitual de discriminar entre empresas activas y fracasadas. Las empresas fracasadas incluyen empresas en suspensión de pagos, en quiebra, absorbidas y disueltas en el período 2000-2003, según la base de datos SABI. Una vez analizado el problema de clasificación dicotómico, se desagrega el fracaso en dos niveles *fracaso1*, que incluye empresas absorbidas y disueltas, y *fracaso2*, que incluye las empresas inmersas en un proceso de suspensión de pagos o quiebra.

2.- **JURIDICA**. La forma jurídica de la empresa es una variable cualitativa que presenta seis modalidades: *Sociedad anónima, Sociedad limitada, Cooperativas, Sociedad comanditaria, Sociedad regular colectiva y Asociación y no definidas*. Sin embargo, las cuatro últimas modalidades son muy minoritarias, por lo que se decidió

agruparlas en una, quedando así esta variable con tres categorías: *Sociedad anónima, Sociedad limitada y Otras.*

3.- **CNAE1**. Partiendo de la CNAE93 a tres dígitos, se realizó un proceso de agrupación con el objetivo de formar grupos de actividades homogéneas, y que al mismo tiempo se redujese el número de categorías de este atributo a un valor razonable. Al final se consigue establecer diez categorías equivalentes al uso de la CNAE93 a un dígito.

4.- Ingresos Netos/ Activo Total **(IN/AT)**. Según puede verse en la ayuda de la base de datos SABI (Informa), se trata de un ratio cinético y como tal es una relación que en lugar de medir únicamente un valor relativo, sirve para analizar una rotación a nivel específico de actividad. Por tanto, refleja las veces que se ha utilizado el total de activo en la obtención de las ventas. Interesa que su valor sea lo más elevado posible, ya que significa un buen aprovechamiento de los recursos disponibles.

5.- Beneficio antes de Intereses e Impuestos/ Activo Total **(BAII/AT)**. Según puede verse en Rivero y Banegas (1998), este ratio recoge la *rentabilidad económica* de la empresa pues trata de medir el rendimiento de las inversiones, o sea, el grado de aprovechamiento de los activos y el nivel de eficiencia en el desarrollo de las funciones operativas. Es decir, informa de la capacidad de la estructura económica para generar beneficios, independientemente de cómo se encuentre financiada la empresa.

6.- Beneficio antes de Intereses e Impuestos/ Fondos Propios **(BAII//FP)**. Este ratio es, en cierto modo, complementario del anterior ya que recoge la *rentabilidad financiera* de la empresa. Para ello da información referente a la rentabilidad media obtenida por la empresa mediante su actividad, a partir de los fondos propios. Dicho de otra forma, este ratio informa acerca de la eficiencia de la empresa en la utilización de los fondos aportados por los accionistas.

7.- Activo Circulante/Activo Total **(AC/AT)**. Este es un ratio de estructura económica, que recoge el peso que el activo circulante presenta sobre el activo total, por ello se conoce este ratio como *peso del circulante*.

8.- Tesorería/ Activo Total **(T/AT)**. Este es un segundo ratio de estructura económica y recoge la importancia que sobre el activo total tiene la tesorería. A este ratio se le conoce como *peso de tesorería*.

9.- Ventas / Fondos Propios **(V/FP)**. Según puede verse en Rivero y Banegas (1998), el objetivo de este ratio de estructura financiera es medir el nivel de rotación de los capitales propios en relación a la cifra de ventas. En Rivero Torre (2002) se señala que este ratio es indicativo, en sentido indirecto, de las veces que se han utilizado los recursos propios en la financiación del circulante.

En esta rotación se considera conveniente que el ratio obtenido no sea elevado, ya que ello indicaría una posición financiera equilibrada, mientras que uno elevado mostraría que la empresa está empleando endeudamiento en exceso, y cualquier caída de las ventas podría ocasionar una situación financiera precaria, con incidencia negativa, a su vez, en los resultados económicos.

10.- Activo Circulante/Pasivo Circulante **(AC/PC)**. Este es el *ratio de solvencia* que indica la capacidad de generar recursos financieros líquidos suficientes, por parte de la empresa, para atender puntualmente a sus compromisos de pago y deudas a corto plazo, con los cobros realizables también a corto plazo. Así pues, el activo circulante supone el potencial con que la empresa cuenta para hacer frente a sus obligaciones de pago con vencimiento en un período corto de tiempo. De esta forma, cuanto mayor sea el índice resultante de este ratio, mayor será también la garantía o margen de seguridad otorgado a los acreedores funcionales a corto plazo.

11.- Tesorería / Pasivo Circulante **(T/PC)**. Este ratio se conoce como *test ácido*. Indica la capacidad de respuesta inmediata de la empresa para atender a los

compromisos de pago de las deudas a corto plazo derivadas del ciclo productivo y comercial. Presenta la relación entre la tesorería de la empresa con respecto al pasivo circulante.

Un valor elevado del ratio indicaría exceso de tesorería, lo que perjudica la eficiencia y rentabilidad final, aunque esto dependerá de la actividad concreta de la empresa y de las circunstancias por las que atraviese en ese instante determinado. Por otro lado, un valor muy reducido del ratio, muy inferior a la unidad, implicaría incapacidad para afrontar los compromisos de pago. Luego es preferible un valor intermedio, ni muy alto ni muy bajo.

12.- Fondo de Maniobra/Ventas **(FM/V)**. Este ratio es conocido como peso del fondo de maniobra sobre las ventas. Relaciona el fondo de maniobra, calculado como diferencia entre el activo circulante y el pasivo circulante, con las ventas del ejercicio. Este es un ratio de liquidez cuyo objetivo es medir la liquidez neta de los activos sobre el total de ventas.

13.- Pasivo Exigible/ Pasivo Total **(PE/PT)**. Este es un ratio de solvencia que en SABI se denomina *de endeudamiento,* ya que aporta información referente al peso de las deudas a corto y largo plazo en el total del pasivo. Dependiendo de cuál sea el coste de la financiación ajena y de la financiación propia interesará un valor determinado de este ratio.

14.- Cash Flow/ Pasivo Total **(CF/PT)**. Este ratio mide la capacidad de devolución del pasivo. Representa el volumen de liquidez disponible, resultante de los ingresos del ejercicio económico, una vez pagados todos los gastos exigibles que han sido necesarios para la obtención de aquéllos, para atender al pago de la devolución de los préstamos de la deuda total y de la financiación propia. Cuanto mayor sea este índice, mayor será la capacidad de la empresa para retribuir la financiación.

15.- Fondo de Maniobra/ Activo Total **(FM/AT)**. Este es un ratio de liquidez cuyo objetivo, según Rivero y Banegas (1998), es medir la liquidez neta de los activos sobre el total de capitalización, significando la importancia que representa el fondo de maniobra en relación con el activo total y, por tanto, la garantía que ofrece la empresa con su activo circulante para afrontar un pasivo de igual grado.

16.- Ventas/ Activo Total **(V/AT)**. Señala, de forma figurada, las veces que se ha vendido el activo total y, por tanto, el número de rotaciones del mismo, o sea, cuántas veces se ha vendido y repuesto. También, visto de otro modo, puede interpretarse como los euros vendidos por cada euro invertido en activo, reflejando, de un modo u otro, la capacidad de los activos para generar ventas y la eficiencia relativa con que la empresa los gestiona. Cuanto mayor sea el ratio, si el margen comercial es el mismo, implicará un mayor beneficio para una inversión inferior y, por tanto, un aumento de la rentabilidad.

17.- Ventas/ Activo Circulante **(V/AC)**. Del mismo que el ratio anterior éste es un ratio de eficiencia y productividad, que puede entenderse como los euros vendidos por cada euro invertido en activo circulante, es decir, muestra la capacidad de los activos circulantes para generar ventas y la eficiencia con la que la empresa los gestiona.

18.- Logaritmo del Activo Total **(ln(AT))**. Esta variable que ya utilizaron Frydman y Altman (1985), es una variable que se utiliza para recoger de manera aproximada el tamaño de la empresa, ya que el mismo se considera un factor muy relevante para la predicción del fracaso empresarial. Otras opciones para aproximar el tamaño de la empresa es el número de trabajadores empleados en la misma o el volumen de ventas. A falta de una teoría que diga claramente cuál de estas opciones es mejor, en este trabajo se ha optado por utilizar el logaritmo neperiano del activo total como variable indicativa del tamaño de la empresa, al igual que hicieron López y Gandía (1998).

19.- Logaritmo del Activo Circulante **(ln(AC))**. Esta es la única variable que se ha seleccionado, casi exclusivamente, porque funcionaba bien en los modelos. Al trabajar

con el logaritmo del Activo Total, se pensó por qué no probar también con otras variables en logaritmos, de este modo se comprobó que el logaritmo del activo circulante era útil en los modelos entrenados.

Una vez calculados los ratios se aborda la tarea de *detección de atípicos*. Esta búsqueda de valores atípicos se ha realizado desde una perspectiva multivariante para la cual, como se ha comentado en el apartado 9.2.3.1 de este trabajo, se ha utilizado la distancia de Mahalanobis de las observaciones al centro de sus respectivas clases. Para este cálculo las variables cualitativas (CNAE1 y JURIDICA) se han dejado al margen. De esta manera, para cada clase, se eliminan las observaciones que superan un determinado umbral en el valor de su distancia de Mahalanobis, en concreto se eliminan 161 empresas de la clase fracaso1, 1.499 empresas activas y 78 empresas de la clase fracaso2.

Para reducir la presencia de *ruido* se han eliminado aquellas empresas consideradas como activas que hayan presentado resultados negativos en los tres últimos ejercicios, considerando que, aunque son empresas activas, si sufren pérdidas continuadas, pronto entraran en alguna de las modalidades de fracaso, y por tanto, pueden dificultar la labor de aprendizaje del sistema de clasificación. Por este motivo se eliminan 95 empresas activas quedando la muestra finalmente constituida por 853 empresas del tipo fracaso1, 15.174 activas y 512 empresas del tipo fracaso2, lo que supone un total de 16.539 empresas.

Clase	Inicial	Atípicos	Ruido	Final
Fracaso1	1014	161	-	853
Activa	16768	1499	95	15174
Fracaso2	590	78	-	512
Total	18372	1738	95	16539

Tanto en el caso dicotómico como cuando se trabaja con tres clases, se utilizan tamaños iguales para todas las clases, limitado en consecuencia por la clase que menos

observaciones tenga. Para el caso dicotómico se trabaja con 1.365 observaciones para cada clase, utilizándose todas las fracasadas presentes en la muestra inicial y extrayendo una muestra aleatoria de ese tamaño de las empresas activas. El tamaño total de la muestra fue de 2.730 empresas. En el caso de las tres clases la restricción la impone la clase *fracaso2* que cuenta con 512 observaciones, por tanto hay que extraer muestras de ese tamaño para las activas y las empresas del *fracaso1*. En consecuencia, en este caso, el número total de empresas en la muestra es de 1.536.

La elección de tamaños muestrales iguales para las clases, frente a mantener la proporción poblacional, ha sido una de las elecciones más difíciles en la elaboración del presente trabajo. Finalmente se optó por igualar los tamaños de las clases en la muestra a pesar de los inconvenientes que presenta seleccionar una muestra de esta manera, entre los cuales Palepu (1986), cita tres:

<Sobreestimación de la capacidad predictiva de los modelos.

<El modelo es difícilmente generalizable para el resto de la población.

<Existe la posibilidad de añadir dificultad a la interpretación económica de los modelos.

Pero también existen argumentos a favor de igualar las proporciones de las clases en la muestra, entre ellos Rodríguez (2001b) destaca el hecho de que, en la predicción del fracaso empresarial las proporciones poblacionales son mucho más favorables a las empresas activas, lo que daría lugar, para un mismo tamaño muestral, a un número de empresas fracasadas en la muestra muy reducido si se lleva a cabo el muestreo de forma totalmente aleatoria. En este caso la información contenida en la muestra sería muy limitada y, por tanto, la estimación de los modelos por el procedimiento aleatorio puede llevar a estimadores sesgados.

Además, la verdadera proporción poblacional entre empresas fracasadas y activas no es fácil de calcular en la práctica. Por último, tal y como apunta Rodríguez (2001a) si se considera, que la probabilidad *a priori* de las empresas activas es muy superior a la

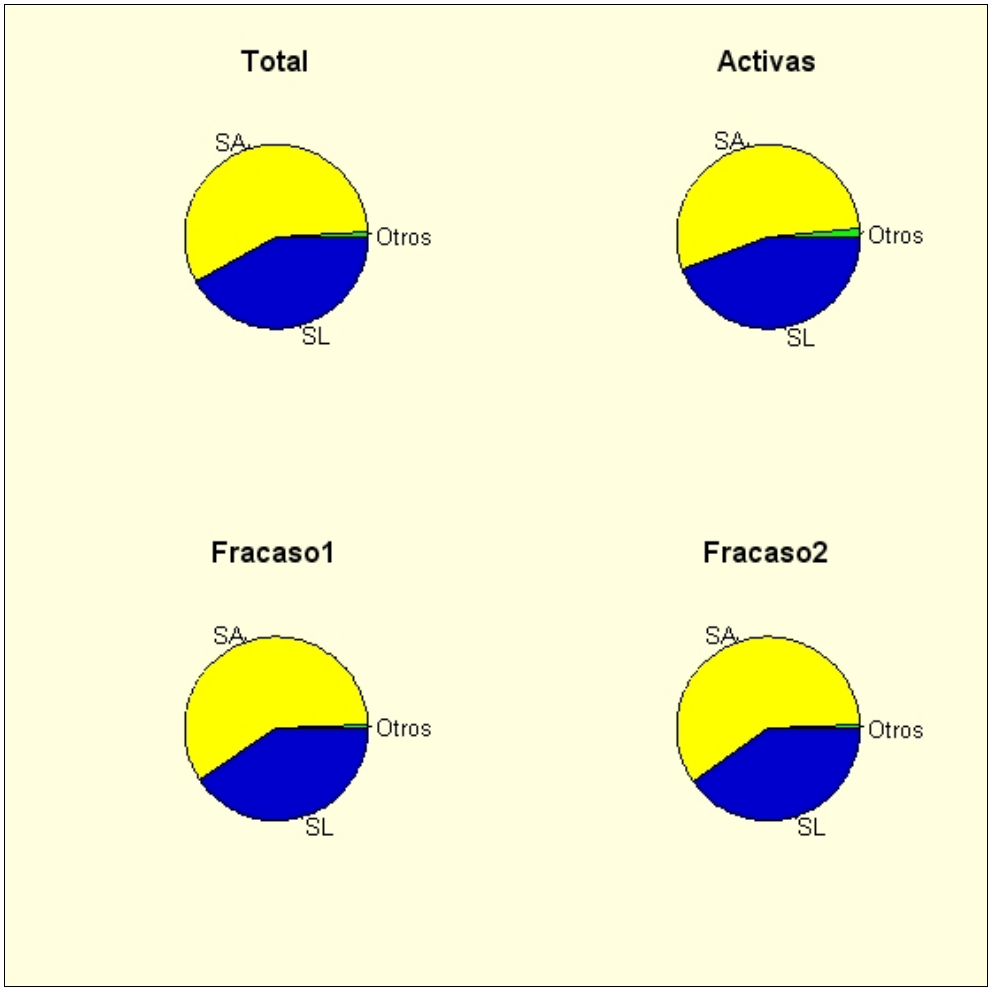
de las empresas fracasadas, ya sea en la muestra o de forma explícita, hay que considerar también los costes de error de tipo I y tipo II, es decir, los costes de clasificar una empresa activa como fracasada, o viceversa. Obviamente, es mucho mayor el coste de clasificar una empresa fracasada como activa. Por tanto, la desproporción existente entre probabilidades a priori y costes de error para cada tipo de empresa puede compensarse y producir un efecto conjunto casi neutro.

Por todo esto, como ya se ha dicho, finalmente se optó por trabajar con el mismo número de empresas para las distintas clases. Sin embargo, no se trata de un emparejamiento como el que se realiza habitualmente en los trabajos de predicción del fracaso, ya que de ese modo se elimina el efecto del tamaño y sector como variables potencialmente discriminantes en el pronóstico. En este trabajo el tamaño de la empresa y el sector sí se recogen como variables discriminantes en las variables lnAT y CNAE1.

A continuación se muestra un resumen de los estadísticos básicos correspondientes a las variables finalmente seleccionadas.

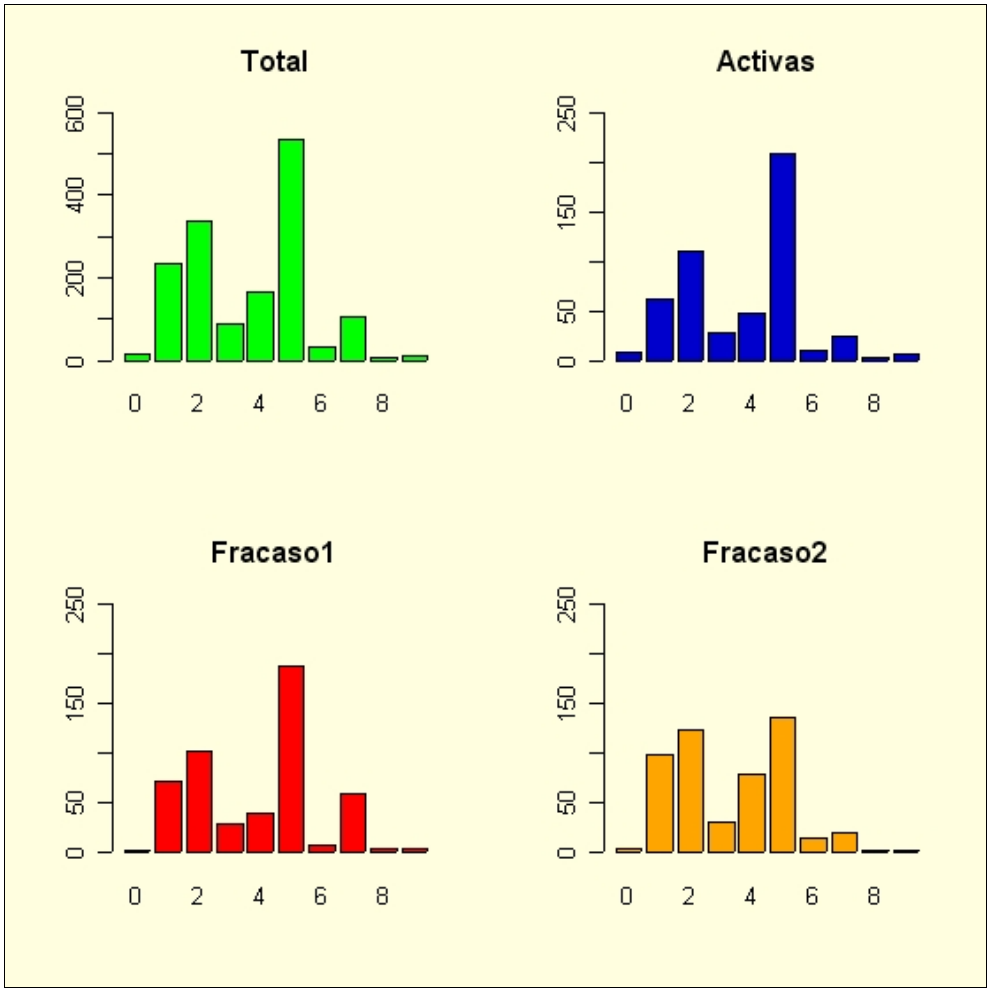
, JURIDICA.

Juridica	total		Activas		Fracaso1		Fracaso2	
	Nº	%	Nº	%	Nº	%	Nº	%
SA	880	57,29	278	54,30	300	58,59	302	58,98
SL	642	41,80	227	44,34	208	40,63	207	40,43
Otros	14	0,91	7	1,37	4	0,78	3	0,59



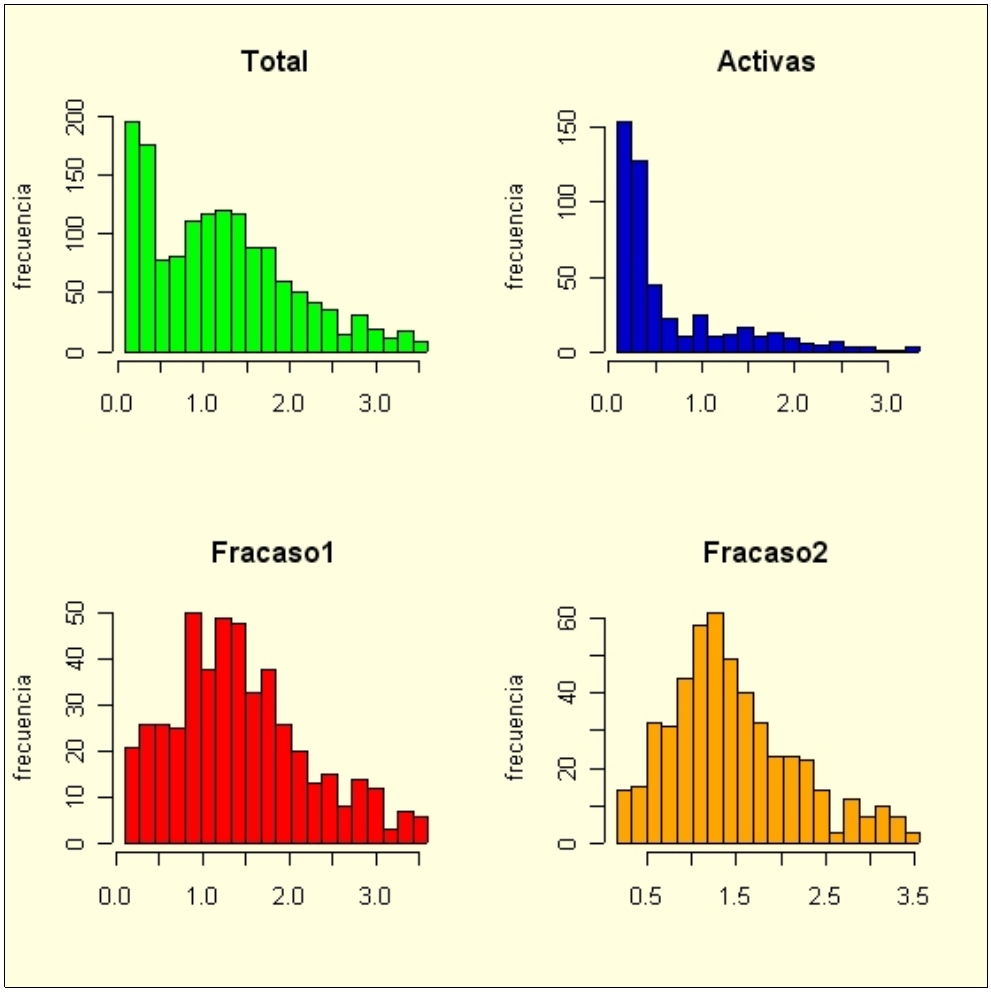
, CNAE1 (Clasificación de la CNAE a un dígito)

CNAE1	total		Activas		Fracaso1		Fracaso2	
	Nº	%	Nº	%	Nº	%	Nº	%
0	16	1,04	9	1,76	3	0,59	4	0,78
1	233	15,17	63	12,30	72	14,06	98	19,14
2	338	22,01	111	21,68	103	20,12	124	24,22
3	89	5,79	29	5,66	29	5,66	31	6,05
4	166	10,81	47	9,18	40	7,81	79	15,43
5	533	34,70	208	40,63	188	36,72	137	26,76
6	32	2,08	10	1,95	8	1,56	14	2,73
7	106	6,90	25	4,88	60	11,72	21	4,10
8	9	0,59	3	0,59	4	0,78	2	0,39
9	14	0,91	7	1,37	5	0,98	2	0,39



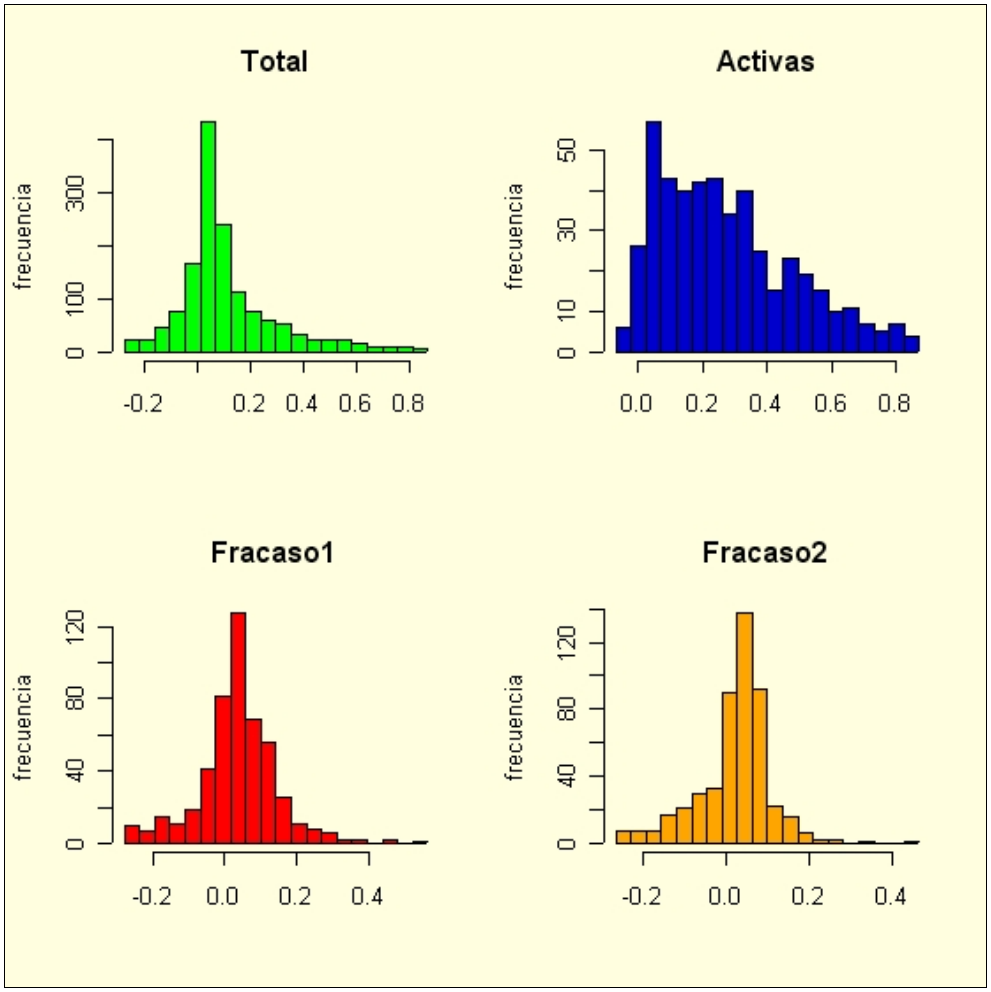
, IN/AT1 (Ingresos Netos/Activo Total)

IN.AT1	Total	Activas	Fracaso1	Fracaso2
Media	1,25083	0,66120	1,58735	1,50394
Mediana	1,10637	0,32558	1,36502	1,34542
Desv tip	1,08635	0,92720	1,23016	0,80381
Asimetria	3,54350	5,19127	4,62029	1,04743
Curtosis	35,75043	44,52333	49,14547	1,31999
Minimo	0,00655	0,00655	0,01084	0,03540
Maximo	17,09422	11,09404	17,09422	4,60959
Primer Cuartil	0,40439	0,18336	0,86084	0,96879
Tercer Cuartil	1,72450	0,84374	1,99181	1,90269
ShapiroWilks	0,79612	0,56894	0,74345	0,93709
SWpvalor	3,45709E-40	1,27305E-33	2,16773E-27	7,00918E-14
KolmogorovSmirnov	0,50952	0,50576	0,58725	0,64437
KSpvalor	0	0	0	0



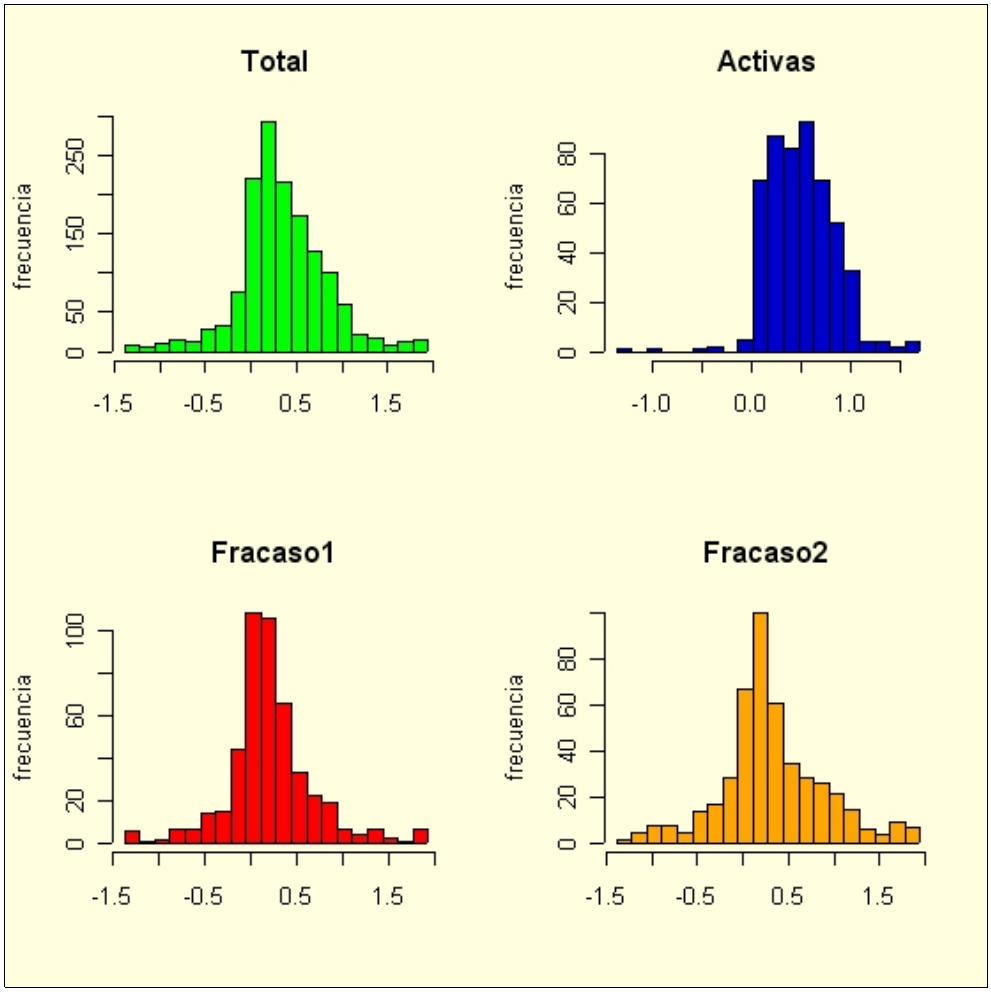
, BAI/AT1 (Beneficios antes de Intereses e Impuestos/Activo Total)

BALAT1	Total	Activas	Fracaso1	Fracaso2
Media	0,15261	0,44154	0,01183	0,00445
Mediana	0,06072	0,25987	0,03393	0,03441
Desv tip	0,69197	1,12447	0,18092	0,12525
Asimetria	16,12711	10,52449	-3,47737	-2,11998
Curtosis	332,45076	130,06576	23,23992	8,37829
Minimo	-1,65987	-0,83158	-1,65987	-0,73550
Maximo	16,82906	16,82906	0,56327	0,46713
Primer Cuartil	0,01017	0,11545	-0,02209	-0,02124
Tercer Cuartil	0,17183	0,47067	0,09269	0,06856
ShapiroWilks	0,25507	0,25770	0,72453	0,80989
SWpvalor	2,48694E-61	7,19887E-41	3,31931E-28	4,25627E-24
KolmogorovSmirnov	0,38045	0,48582	0,37250	0,41133
KSpvalor	0	0	0	0



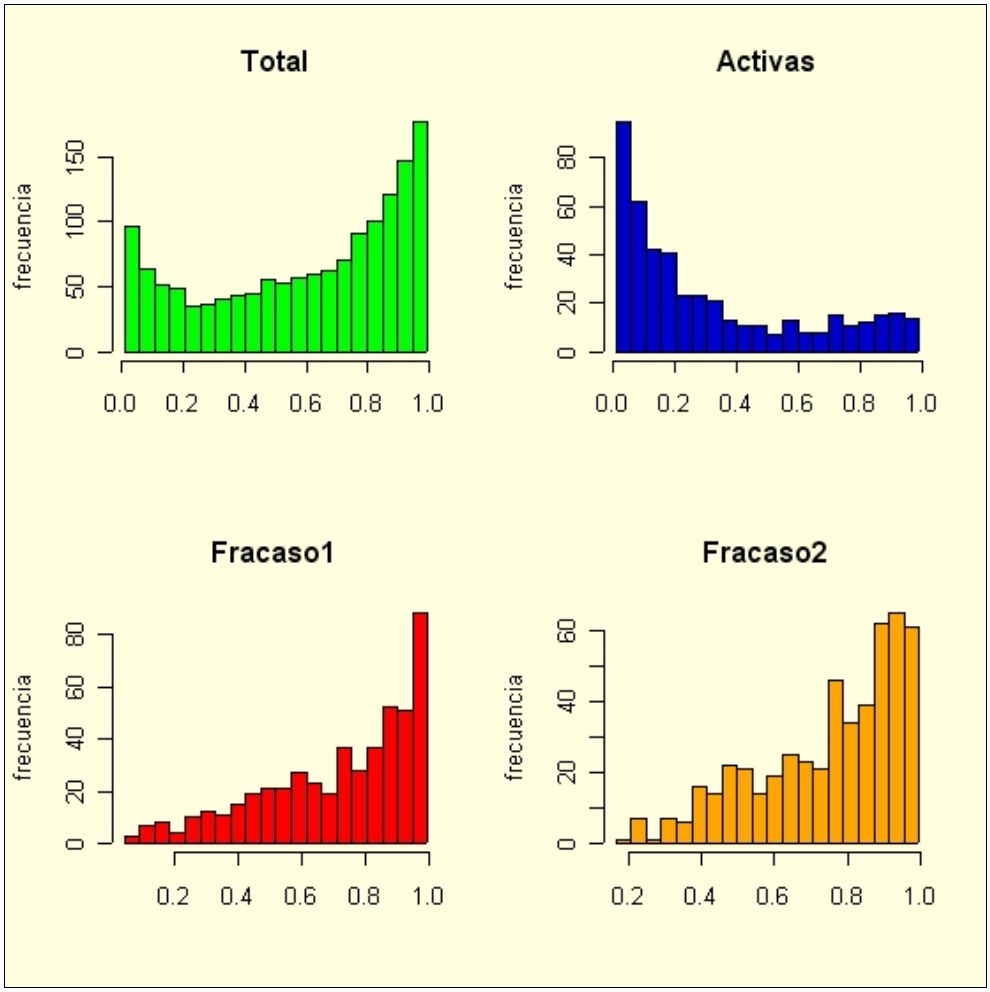
, BAI/FP1 (Beneficios antes de Intereses e Impuestos/Fondos Propios)

BALFP1	Total	Activas	Fracaso1	Fracaso2
Media	0,33157	0,50161	0,17720	0,31590
Mediana	0,29186	0,48865	0,16053	0,25337
Desv tip	1,11161	0,35645	1,25472	1,39908
Asimetria	-1,00194	0,42902	0,95883	-1,93115
Curtosis	36,61884	3,72120	27,65036	24,66094
Minimo	-12,38710	-1,37344	-9,52315	-12,38710
Maximo	9,40564	2,11916	9,40564	7,42857
Primer Cuartil	0,07048	0,25537	-0,03500	0,00000
Tercer Cuartil	0,63939	0,70168	0,38262	0,69492
ShapiroWilks	0,61796	0,94581	0,59326	0,68125
SWpvalor	7,36103E-50	9,79775E-13	6,84969E-33	6,47016E-30
KolmogorovSmirnov	0,32761	0,48517	0,27752	0,27162
KSpvalor	0	0	0	0



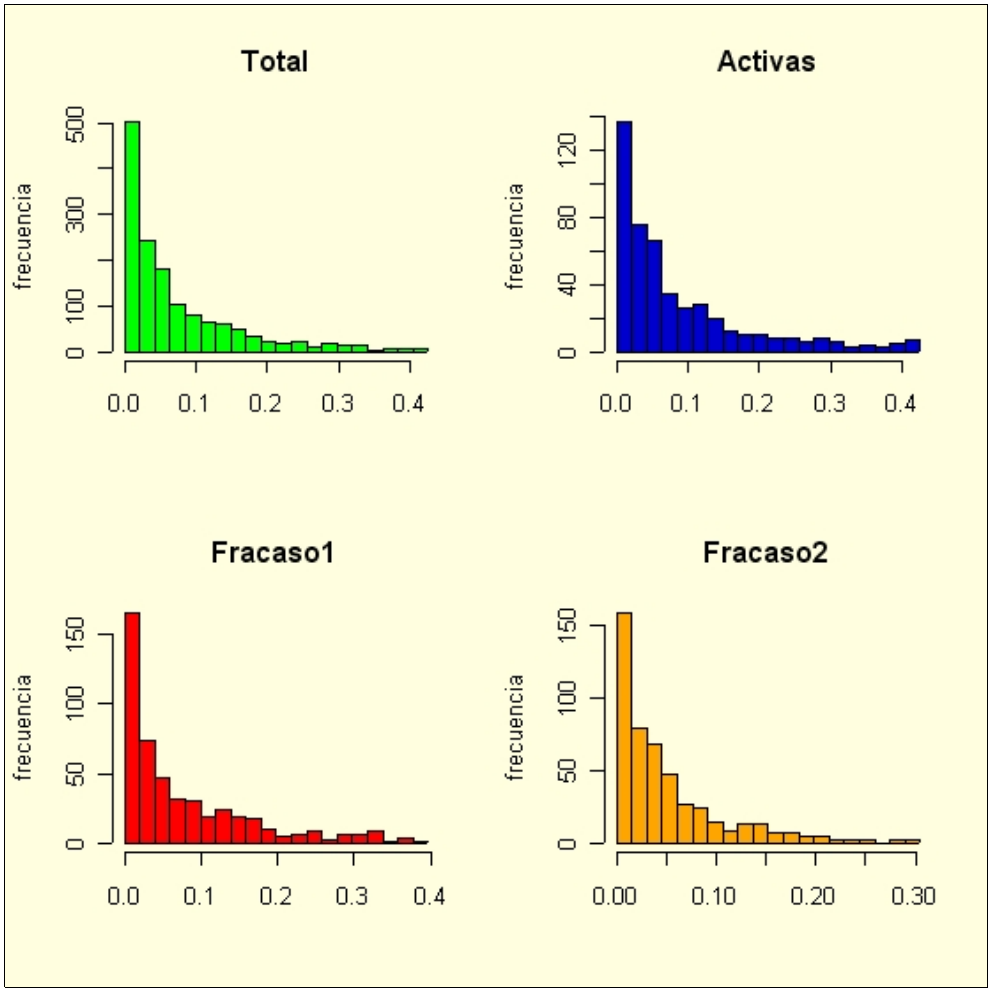
, AC/AT1 (Activo Circulante/Activo Total)

AC.AT1	Total	Activas	Fracaso1	Fracaso2
Media	0,61718	0,37373	0,72431	0,75349
Mediana	0,68549	0,17207	0,79438	0,80831
Desv tip	0,63325	1,00952	0,23938	0,19659
Asimetria	18,33624	14,29279	-0,80263	-0,78022
Curtosis	496,30342	242,33289	-0,32348	-0,37574
Minimo	0,00000	0,00000	0,04444	0,16617
Maximo	18,99798	18,99798	1,00000	1,00000
Primer Cuartil	0,31717	0,05204	0,56172	0,62150
Tercer Cuartil	0,88760	0,50814	0,92634	0,91493
ShapiroWilks	0,36509	0,23095	0,90609	0,91518
SWpvalor	1,78761E-58	2,28837E-41	3,12226E-17	2,42243E-16
KolmogorovSmirnov	0,50000	0,50000	0,55442	0,60638
KSpvalor	0	0	0	0



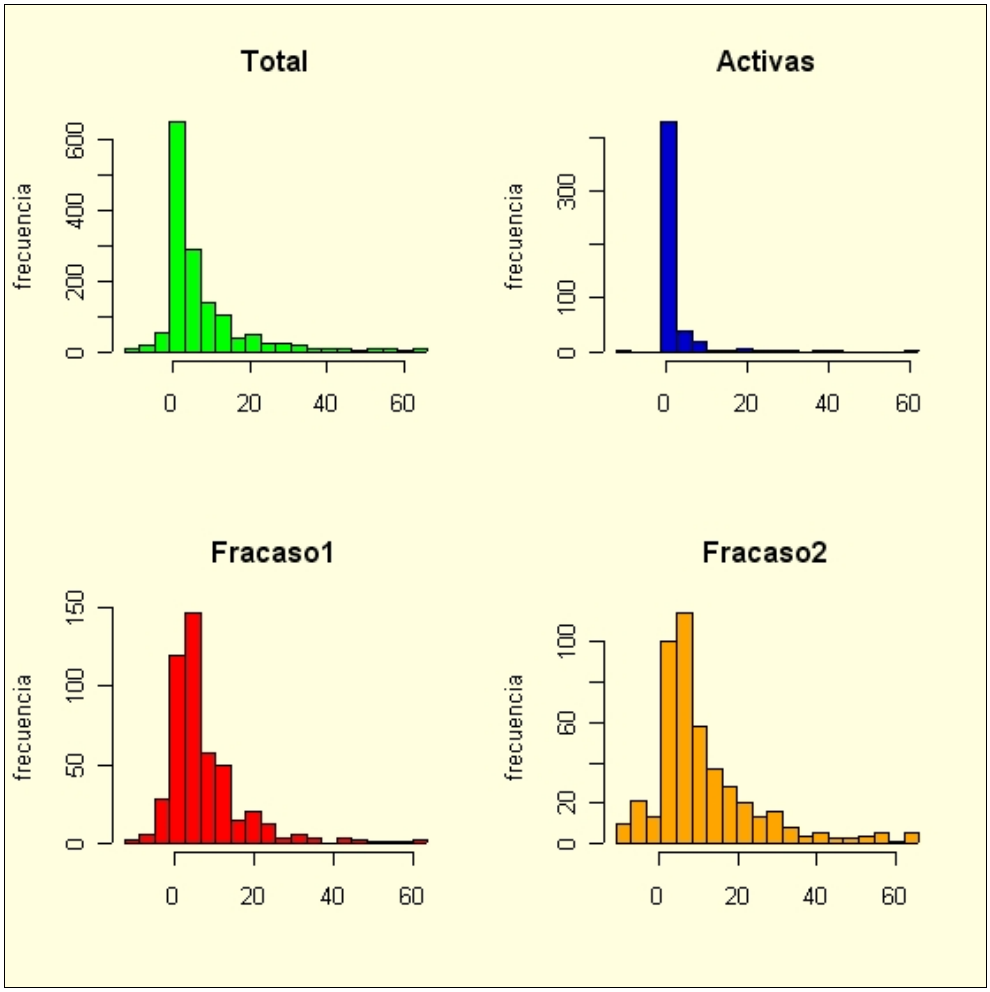
, T/AT1 (Tesorería/Activo Total)

T.AT1	Total	Activas	Fracaso1	Fracaso2
Media	0,08763	0,12457	0,08850	0,04980
Mediana	0,04084	0,05165	0,04315	0,03037
Desv tip	0,14894	0,21592	0,11812	0,05710
Asimetria	5,92103	4,88258	2,65337	1,84626
Curtosis	59,32649	34,61709	10,03742	3,52597
Minimo	0,00000	0,00000	0,00000	0,00000
Maximo	2,21501	2,21501	0,93120	0,30469
Primer Cuartil	0,01180	0,01852	0,01232	0,00945
Tercer Cuartil	0,10615	0,13882	0,12330	0,06522
ShapiroWilks	0,53171	0,53697	0,71398	0,78537
SWpvalor	3,16011E-53	1,5597E-34	1,21824E-28	2,1024E-25
KolmogorovSmirnov	0,50000	0,50000	0,50000	0,50000
KSpvalor	0	0	0	0



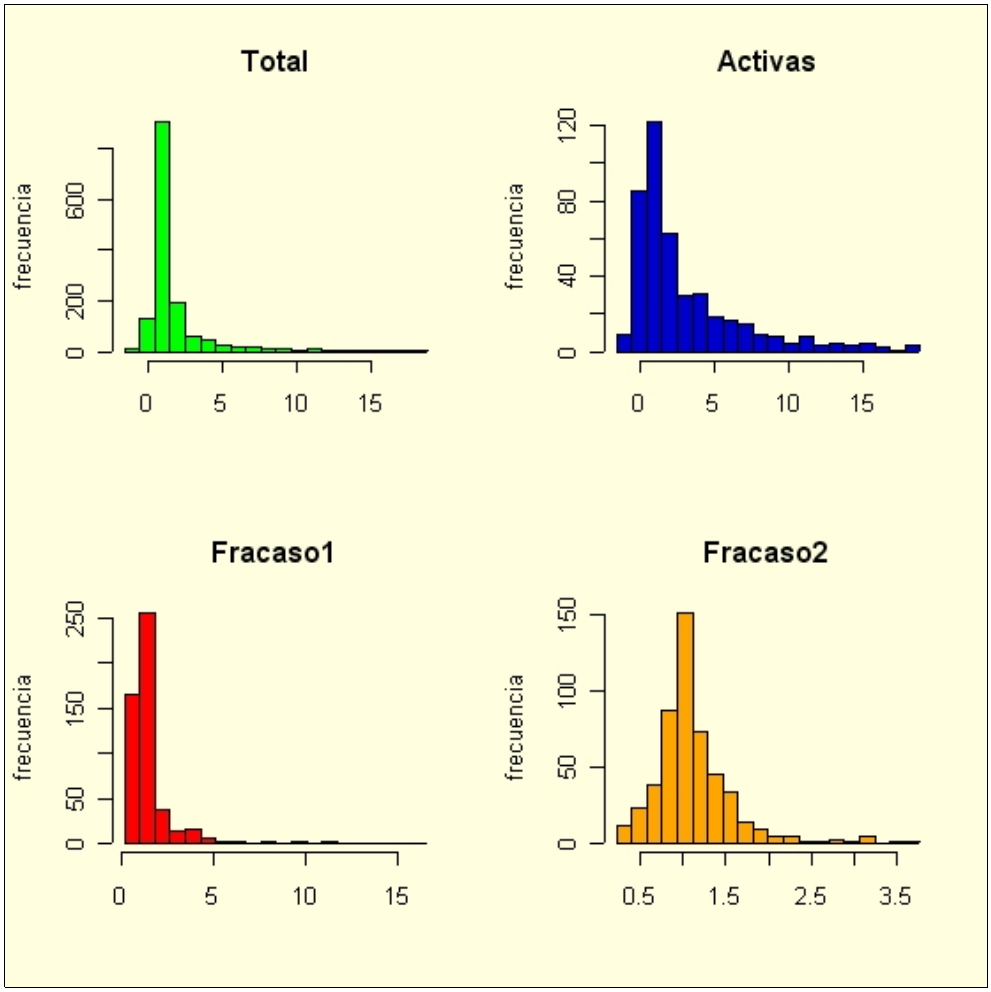
, V/FP1 (Ventas/Fondos Propios)

V.FP1	Total	Activas	Fracaso1	Fracaso2
Media	8,05256	1,55688	9,93965	12,66114
Mediana	3,05195	0,07922	4,68902	7,05141
Desv tip	21,36870	5,88338	23,61606	26,68695
Asimetria	2,92109	3,86041	2,46559	2,11036
Curtosis	24,99952	35,57450	18,64084	16,04262
Minimo	-159,06250	-32,80080	-146,51896	-159,06250
Maximo	195,15385	62,12577	176,33762	195,15385
Primer Cuartil	0,08087	0,03562	1,70793	2,84656
Tercer Cuartil	9,34532	0,19527	11,19563	16,18618
ShapiroWilks	0,58281	0,39451	0,59134	0,70124
SWpvalor	2,65915E-51	4,5739E-38	5,97957E-33	3,77059E-29
KolmogorovSmirnov	0,54445	0,44970	0,70877	0,78124
KSpvalor	0	0	0	0



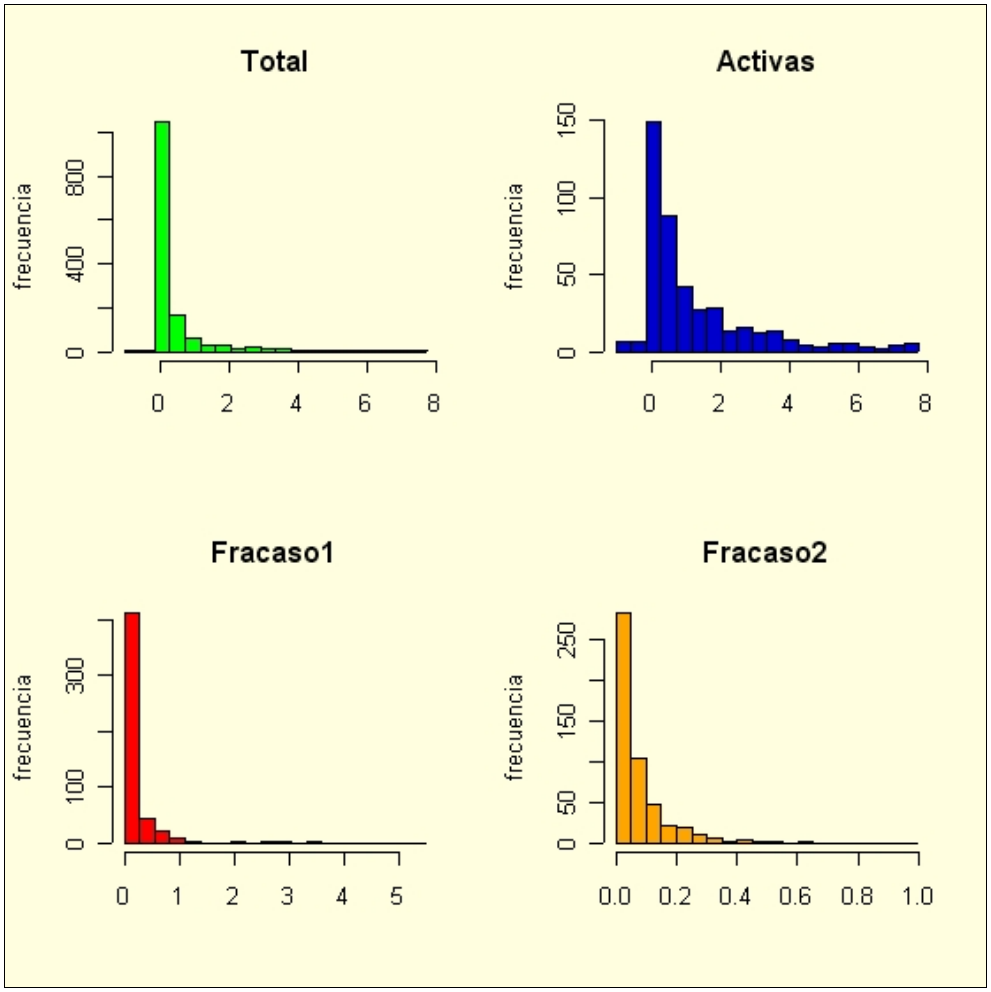
, AC/PC1 (Activo Circulante/Pasivo Circulante)

AC.PC1	Total	Activas	Fracaso1	Fracaso2
Media	11,63648	3,72437	30,06726	1,11783
Mediana	1,12468	1,52585	1,14848	1,04945
Desv tip	343,99941	25,24663	595,24649	0,46539
Asimetria	38,66697	2,08872	22,29666	1,96836
Curtosis	1504,73097	44,01020	498,76438	6,78702
Minimo	-185,49443	-185,49443	0,17988	0,21662
Maximo	13431,01563	255,10308	13431,01563	3,77641
Primer Cuartil	0,84638	0,48029	0,92669	0,86891
Tercer Cuartil	1,69327	4,97144	1,54869	1,27547
ShapiroWilks	0,01369	0,41525	0,02426	0,85142
SWpvalor	1,61656E-66	1,34747E-37	8,54468E-45	1,476E-21
KolmogorovSr	0,58481	0,48666	0,65640	0,65263
KSpvalor	0	0	0	0



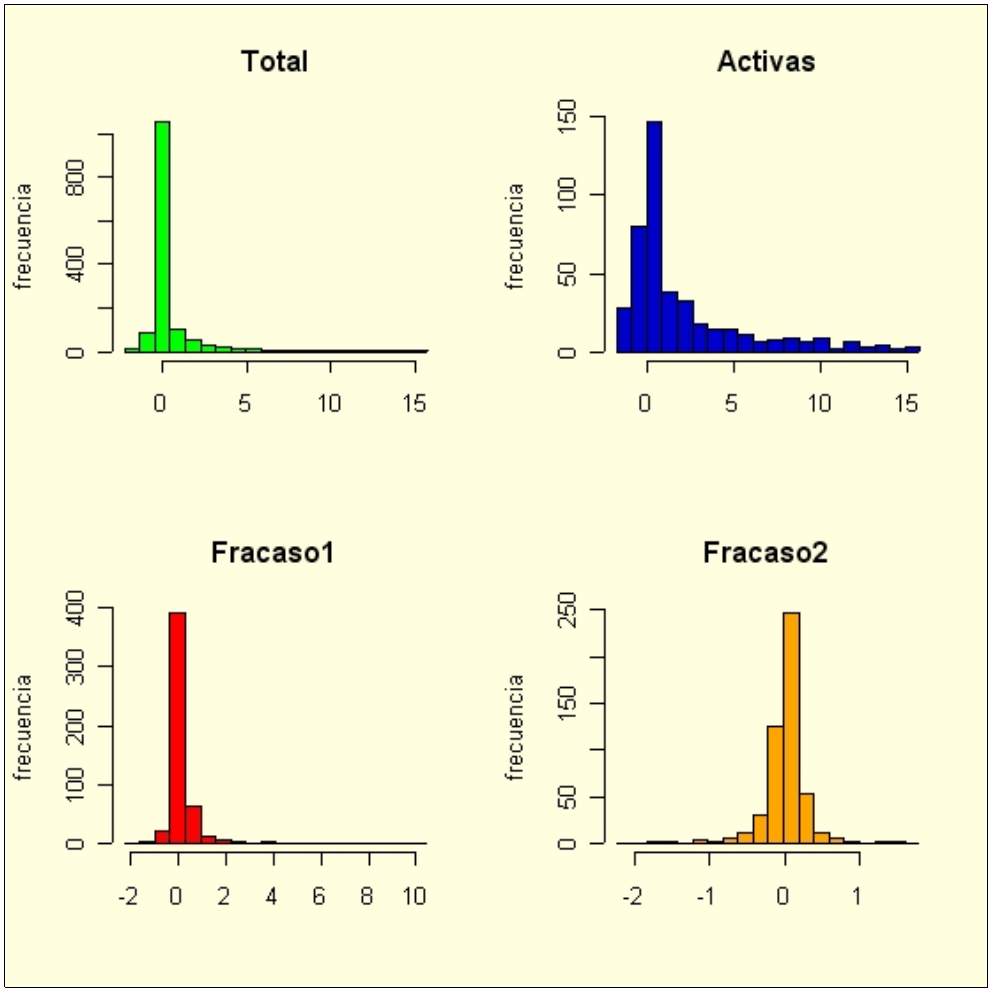
, T/PC1 (Tesorería/Pasivo Circulante)

T.PC1	Total	Activas	Fracaso1	Fracaso2
Media	3,21268	1,52468	8,03375	0,07961
Mediana	0,08590	0,51932	0,07493	0,04165
Desv tip	100,78648	7,52604	174,41579	0,11176
Asimetria	39,00190	0,88291	22,49370	3,24627
Curtosis	1523,08255	25,63477	504,97238	15,22024
Minimo	-49,02778	-49,02778	0,00000	0,00000
Maximo	3946,82813	67,46581	3946,82813	1,00136
Primer Cuartil	0,01894	0,08371	0,01856	0,01297
Tercer Cuartil	0,38038	2,03934	0,22885	0,09923
ShapiroWilks	0,01340	0,56464	0,02161	0,66113
SWpvalor	1,59581E-66	9,52673E-34	7,79395E-45	1,19581E-30
KolmogorovSmirnov	0,46549	0,39648	0,50000	0,50000
KSpvalor	0	0	0	0



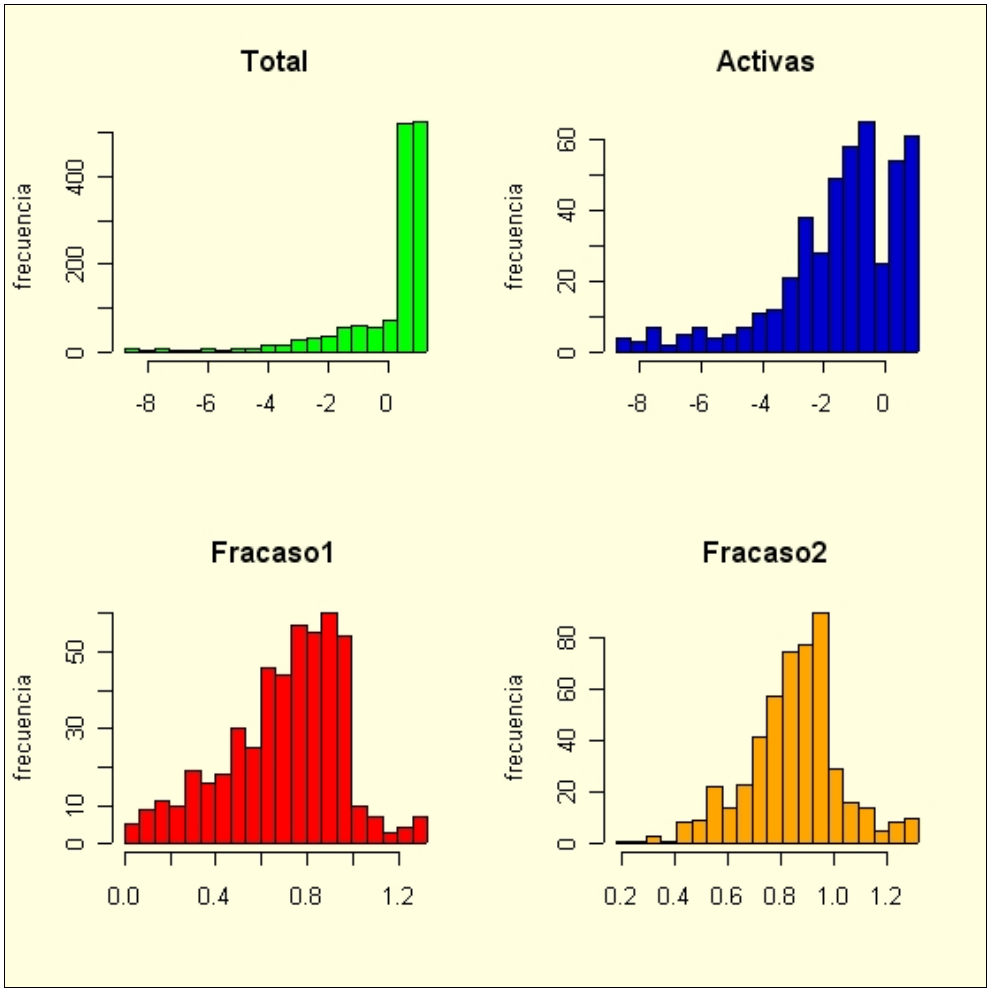
, FM/V1 (Fondo de Maniobra/Ventas)

FM.V1	Total	Activas	Fracaso1	Fracaso2
Media	1,31706	3,61059	0,35901	-0,01841
Mediana	0,06591	0,39966	0,05549	0,02287
Desv tip	7,65242	12,25854	4,17177	0,50522
Asimetria	5,86995	2,92172	20,93984	-6,73459
Curtosis	89,49377	30,81894	457,35390	79,56150
Minimo	-83,63629	-83,63629	-4,60981	-6,67692
Maximo	128,21540	128,21540	92,27279	1,80901
Primer Cuartil	-0,05540	-0,10156	-0,03678	-0,06606
Tercer Cuartil	0,32836	3,88115	0,21695	0,11079
ShapiroWilks	0,32088	0,57477	0,08126	0,50868
SWpvalor	1,14188E-59	1,89211E-33	6,48445E-44	2,67548E-35
KolmogorovSr	0,26671	0,32491	0,31520	0,31221
KSpvalor	0	0	0	0



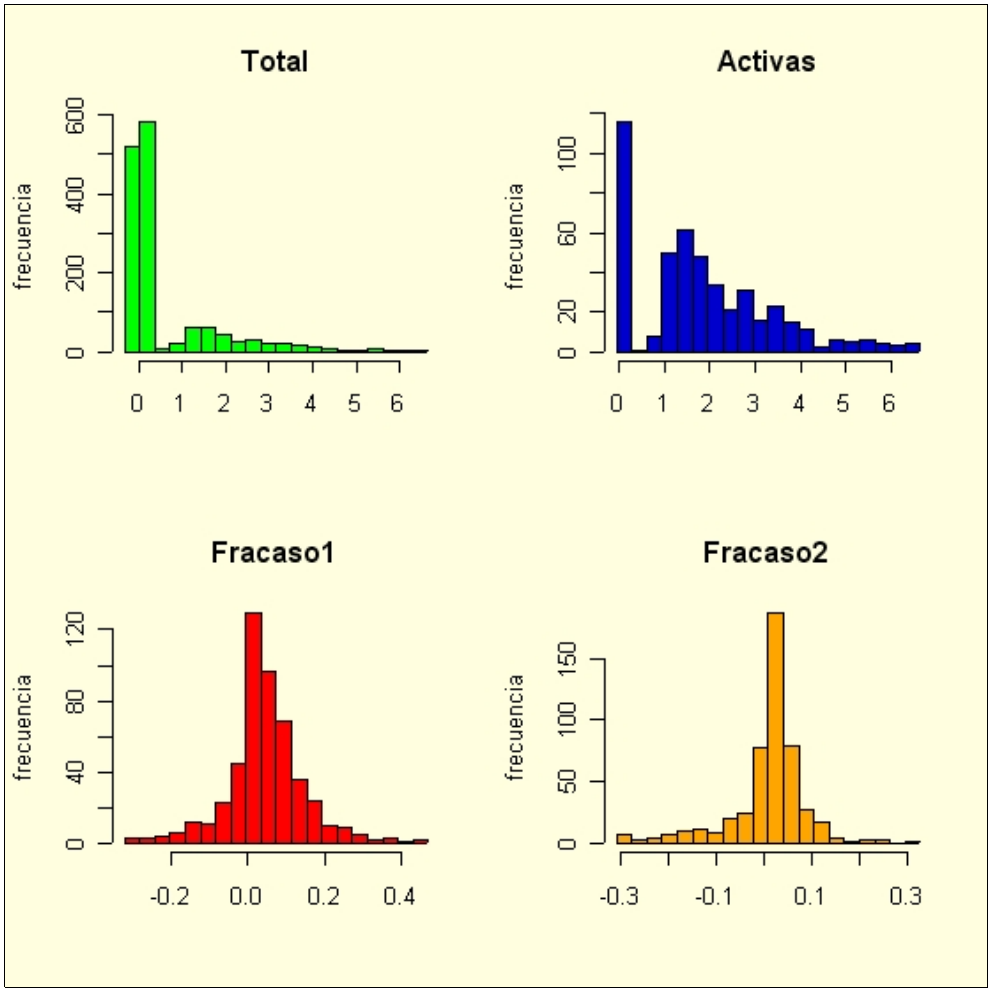
, PE/PT1 (Pasivo Exigible/Pasivo Total)

PE.PT1	Total	Activas	Fracaso1	Fracaso2
Media	-0,26534	-2,40315	0,74734	0,85980
Mediana	0,70935	-1,22955	0,75470	0,86702
Desv tip	3,46505	5,38713	0,35935	0,20302
Asimetria	-7,39694	-4,67424	2,12392	0,60257
Curtosis	105,80186	44,47526	12,91494	3,24770
Minimo	-62,22449	-62,22449	0,00008	0,17696
Maximo	23,13808	23,13808	3,68178	1,98666
Primer Cuartil	-0,07792	-2,83524	0,54901	0,75367
Tercer Cuartil	0,89236	-0,08201	0,90264	0,95182
ShapiroWilks	0,41891	0,58921	0,85353	0,95039
SWpvalor	6,38928E-57	5,14535E-33	2,05165E-21	4,39992E-12
KolmogorovSmirnov	0,34608	0,39748	0,53974	0,65816
KSpvalor	0	0	0	0



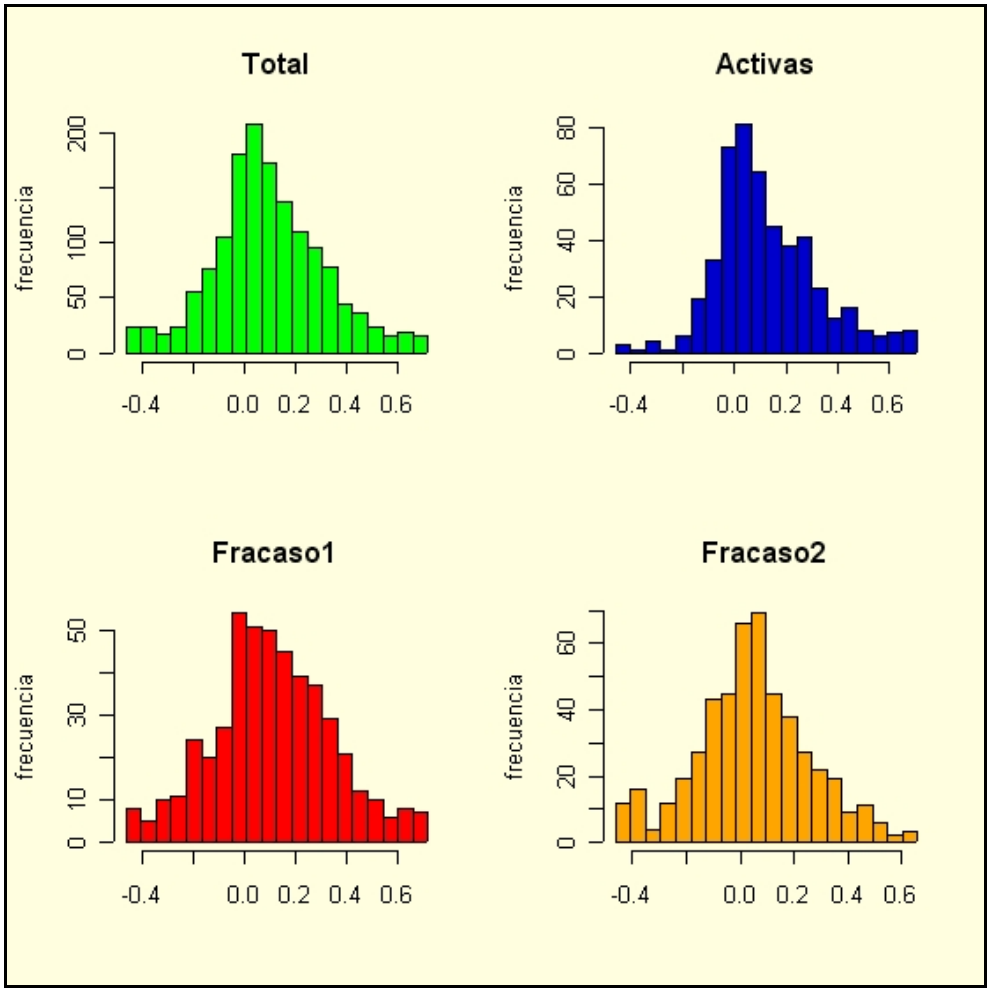
, CF/PT1 (Cash Flow/Pasivo Total)

CF.PT1	Total	Activas	Fracaso1	Fracaso2
Media	0,86719	2,58873	0,01755	-0,00470
Mediana	0,05079	1,68158	0,03488	0,01918
Desv tip	2,97177	4,69228	0,20753	0,11750
Asimetria	8,78817	5,56399	-6,50855	-2,46950
Curtosis	145,28351	59,57837	72,33577	9,93371
Minimo	-21,34693	-21,34693	-2,77353	-0,69850
Maximo	61,05831	61,05831	0,47032	0,32861
Primer Cuartil	0,01068	0,78912	-0,00258	-0,01218
Tercer Cuartil	0,78202	3,09434	0,09213	0,04373
ShapiroWilks	0,36379	0,52906	0,56803	0,74773
SWpvalor	1,64467E-58	9,44784E-35	1,19663E-33	3,36196E-27
KolmogorovSmirnov	0,38287	0,57728	0,37088	0,42367
KSpvalor	0	0	0	0



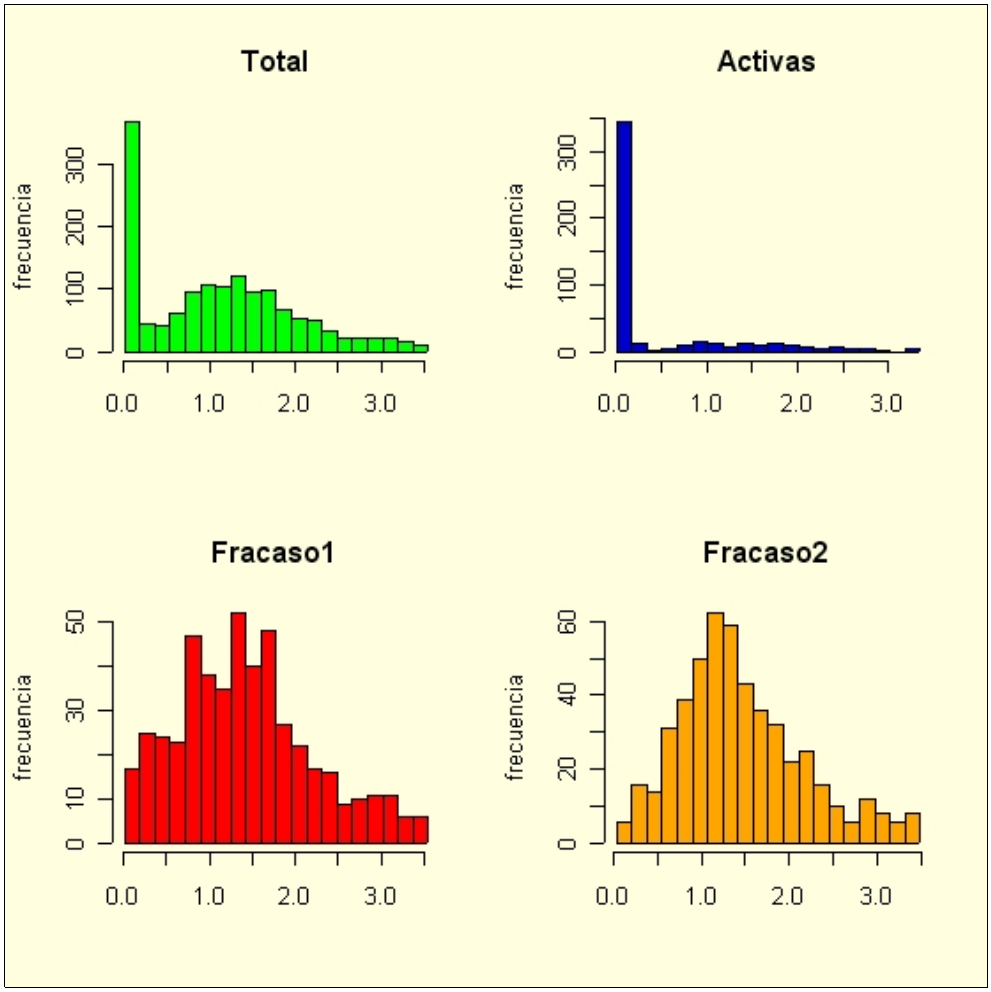
, FM/AT1 (Fondo de Maniobra/Activo Total)

FM.AT1	Total	Activas	Fracaso1	Fracaso2
Media	0,11180	0,21535	0,10308	0,01698
Mediana	0,07253	0,09366	0,09394	0,03398
Desv tip	0,59600	0,93860	0,31718	0,25616
Asimetria	19,48738	14,69895	-0,47123	-0,87510
Curtosis	528,03674	249,32391	2,46536	2,29487
Minimo	-1,53990	-0,61402	-1,53990	-1,08966
Maximo	17,61249	17,61249	0,99993	0,65896
Primer Cuartil	-0,03709	-0,00102	-0,04164	-0,10401
Tercer Cuartil	0,23661	0,25818	0,28274	0,16270
ShapiroWilks	0,33184	0,20243	0,96347	0,95199
SWpvalor	2,22547E-59	6,98822E-42	5,70432E-10	7,60478E-12
KolmogorovSmirnov	0,32887	0,40378	0,31195	0,29875
KSpvalor	0	0	0	0



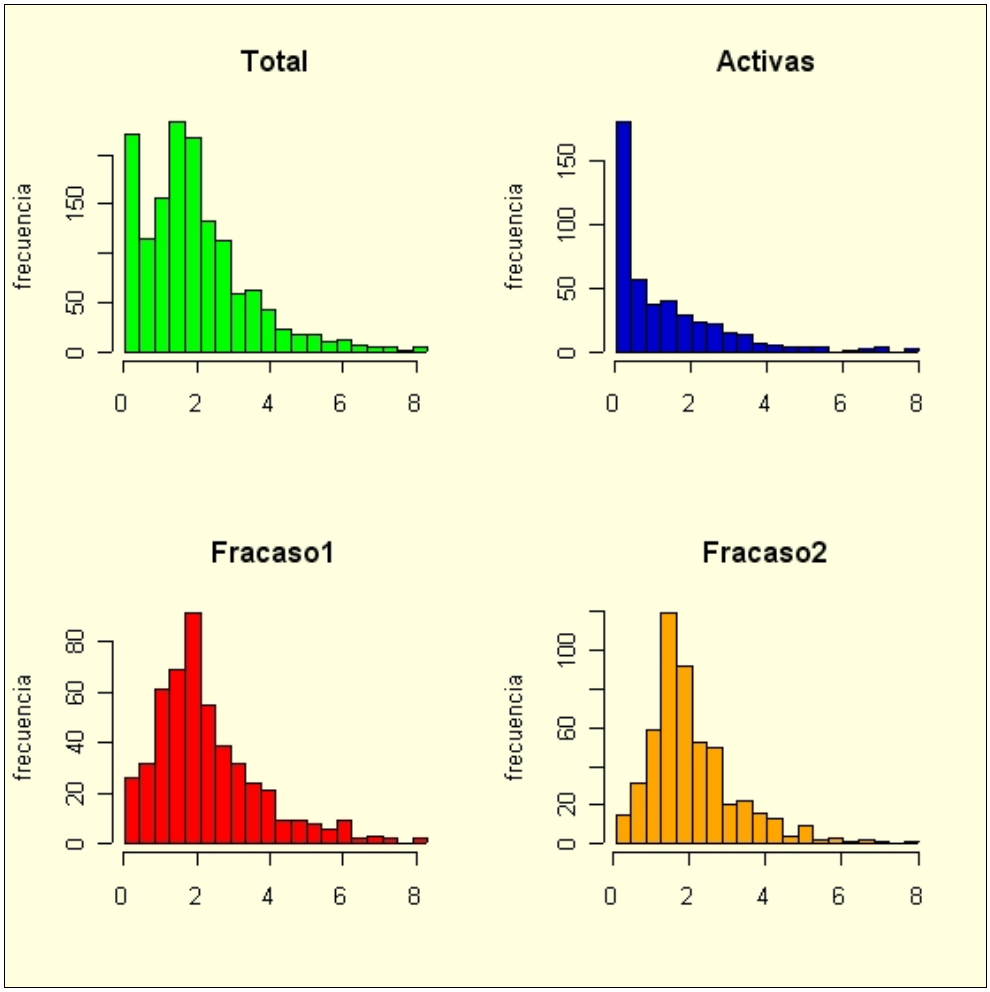
, V/AT1 (Ventas /Activo Total)

V.AT1	Total	Activas	Fracaso1	Fracaso2
Media	1,17350	0,38807	1,61516	1,51727
Mediana	1,08918	0,05528	1,41179	1,35914
Desv tip	1,12826	0,70662	1,32117	0,80444
Asimetria	4,00499	1,99052	6,31570	1,05105
Curtosis	56,72142	3,12655	81,65952	1,33020
Minimo	-0,50643	-0,50643	0,01084	0,03844
Maximo	20,55933	3,35316	20,55933	4,60959
Primer Cuartil	0,14206	0,02033	0,87306	0,98401
Tercer Cuartil	1,72443	0,17167	2,01013	1,91295
ShapiroWilks	0,80007	0,62367	0,67207	0,93684
SWpvalor	6,64746E-40	6,31231E-32	2,96454E-30	6,51551E-14
KolmogorovSmirnov	0,47827	0,45694	0,59619	0,65070
KSpvalor	0	0	0	0



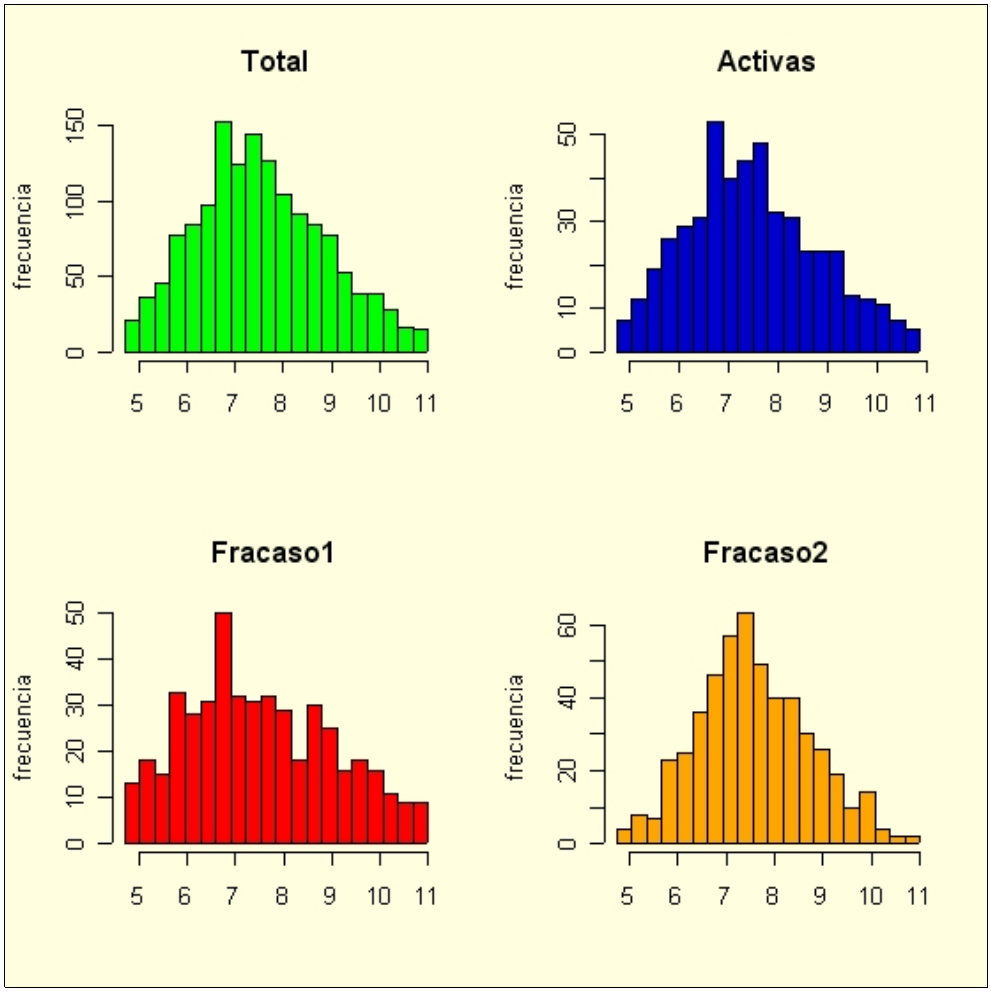
, V/AC1 (Ventas /Activo Circulante)

V.AC1	Total	Activas	Fracaso1	Fracaso2
Media	2,45649	2,81638	2,46245	2,09064
Mediana	1,68232	0,63615	1,97704	1,82037
Desv tip	5,79496	9,75972	2,00766	1,15300
Asimetria	10,10059	6,20421	3,17733	1,45516
Curtosis	133,79243	47,04592	17,97233	2,92746
Minimo	-31,30337	-31,30337	0,01084	0,09163
Maximo	101,30404	101,30404	20,55933	8,02367
Primer Cuartil	0,87854	0,14234	1,28819	1,37403
Tercer Cuartil	2,62607	2,10002	3,04416	2,55985
ShapiroWilks	0,28156	0,31756	0,75478	0,89471
SWpvalor	1,11943E-60	1,06344E-39	7,02531E-27	2,92212E-18
KolmogorovSmirnov	0,56614	0,43851	0,69235	0,73387
KSpvalor	0	0	0	0



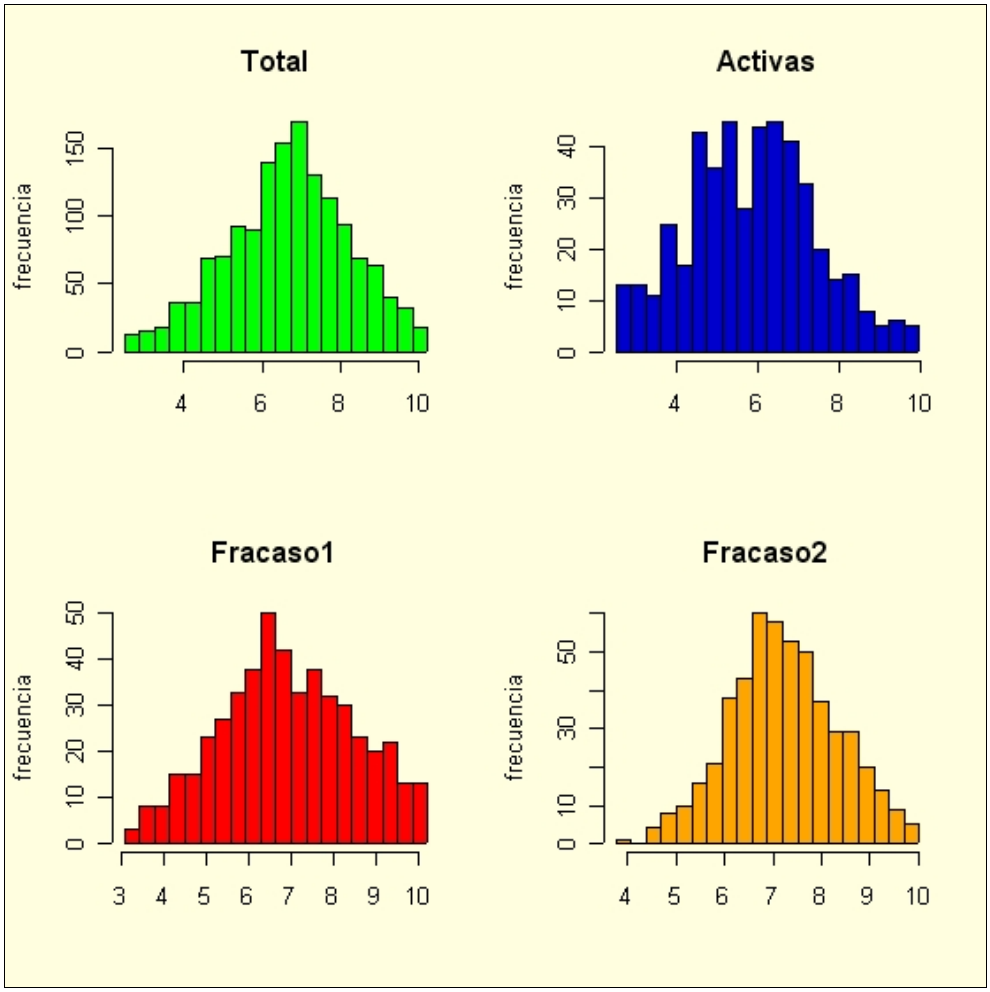
, lnAT1 (logaritmo del Activo Total)

lnAT1	Total	Activas	Fracaso1	Fracaso2
Media	7,56694	7,57157	7,52702	7,60224
Mediana	7,41678	7,40535	7,31726	7,47307
Desv tip	1,62534	1,63212	1,92975	1,24294
Asimetria	0,71436	1,04087	0,56287	0,45642
Curtosis	1,69275	2,72132	0,53416	0,78366
Minimo	3,13675	3,95778	3,13675	3,76120
Maximo	16,56124	16,56124	15,17790	12,21092
Primer Cuartil	6,54858	6,53169	6,16585	6,78700
Tercer Cuartil	8,50334	8,43352	8,76080	8,40301
ShapiroWilks	0,97349	0,95100	0,98131	0,98573
SWpvalor	3,2498E-16	5,40964E-12	3,80257E-06	6,50753E-05
KolmogorovSmirnov	0,99915	0,99996	0,99915	0,99992
KSpvalor	0	0	0	0



, lnAC1 (logaritmo del Activo Circulante)

lnAC1	Total	Activas	Fracaso1	Fracaso2
Media	6,65989	5,58537	7,11915	7,27513
Mediana	6,74017	5,68584	6,91364	7,18349
Desv tip	1,89098	2,05383	1,81511	1,21702
Asimetria	-0,22613	-0,02618	0,32937	0,49538
Curtosis	1,04682	0,86781	-0,00815	1,07603
Minimo	-0,86512	-0,86512	2,29122	3,76120
Maximo	13,54501	13,54501	13,33358	12,12180
Primer Cuartil	5,55475	4,47582	5,86788	6,49563
Tercer Cuartil	7,78507	6,85886	8,22807	7,98221
ShapiroWilks	0,98771	0,98812	0,99226	0,98294
SWpvalor	4,02638E-10	0,000356189	0,009320555	1,03866E-05
KolmogorovSmirnov	0,96863	0,92543	0,99693	0,99992
KSpvalor	0	0	0	0



9.4. APLICACIÓN DEL MÉTODO BOOSTING A LA PREVISIÓN DEL FRACASO EMPRESARIAL

Una vez explicado el proceso de recogida de la información y el análisis previo de los datos que se ha llevado a cabo, se aborda la tarea de aplicar el método boosting al conjunto de datos establecido. En este punto se presentó el problema de que no existía ningún paquete informático que implementase el algoritmo de dicho método, por este motivo hubo que recurrir al uso del programa R, que es un conjunto de programas para manipulación de datos, cálculo y gráficos. Entre otras características dispone de un lenguaje de programación (lenguaje R) bien desarrollado y efectivo. El programa R tiene muchas cosas en común con el programa S-Plus, pero a diferencia de éste es de distribución gratuita.

Algunas técnicas estadísticas están incluidas en el entorno base de R y otras, en cambio, se acompañan en forma de librerías (packages). Junto con R se incluyen ocho librerías, que se pueden llamar *librerías estándar*, y otras muchas están disponibles en internet en CRAN (*http://www.r-project.org*). Este no es el caso del método boosting, por lo que se ha tenido que aprender a programar en lenguaje R, lo cual dada la formación del doctorando no especializada en temas informáticos, ha supuesto un esfuerzo considerable.

En concreto, durante la aplicación práctica, se han utilizado principalmente las versiones R 2.0.0 (septiembre 2004) y R 2.0.1 (noviembre 2004), si bien, las actualizaciones de este programa se siguen produciendo. Para la aplicación también se utilizan otros sistemas de clasificación como los árboles de clasificación, que sí se encuentran disponibles mediante librerías en la página web antes indicada. En concreto, para construir los árboles se ha utilizado la librería *rpart*, para el discriminante lineal la librería *MASS*, para el vecino más próximo la librería *class*, para el bosque aleatorio la librería *randomForest* y, por último, en el caso de las redes neuronales se ha utilizado la librería llamada *nnet*. Cada una de estas funciones tiene una serie de parámetros a ajustar cuyo significado se puede conocer a través de la ayuda que cada librería posee.

La estructura de la aplicación es la siguiente:

< En primer lugar, se trabaja con el caso dicotómico, utilizando sólo la información del último año anterior al fracaso, se aplica inicialmente un árbol de clasificación y, posteriormente, se estudia si el clasificador boosting consigue mejorar la precisión del árbol individual.

< En segundo lugar se repite el proceso, distinguiendo entre tres clases (fracaso1, activas, fracaso2).

< En tercer lugar se compara el método boosting con algunos de los sistemas de clasificación más comúnmente utilizados.

< Por último, se prueba el modelo entrenado con la información del último año antes del fracaso (*t-1*) en el resto de años previos, hasta el quinto año anterior al fracaso (*t-5*) y también se construye un modelo con la información de todos los años previos al fracaso.

9.4.1. Predicción del fracaso empresarial con árboles de clasificación. El caso de dos clases

En primer lugar se construye un árbol que se ajuste perfectamente al conjunto de entrenamiento, formado por 2.456 empresas (90%). Para ello se tendrán que ajustar aquellos parámetros de la función *rpart* que delimitan el crecimiento del árbol. Es de esperar que este árbol, aunque clasifique perfectamente el conjunto de entrenamiento, no tenga un buen comportamiento en el conjunto de prueba (274 empresas, el 10%), por lo que tendrá que ser podado.

```
> sabi.rpart<-rpart(ESTADO~.,method="class", data=sabi[ind,], cp=0,
minsplit=2, maxdepth=30)
```

El error en el conjunto de entrenamiento es cero, ya que el árbol consigue un ajuste perfecto en él, como se puede ver en la matriz de confusión siguiente.

```
>    sabi.predrpart<-predict(sabi.rpart,  newdata  =  sabi[ind,],
type="class")
>  table(sabi.predrpart, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
              Clase real
Clase estimada Fracaso Activa
       Fracaso 1228       0
       Activa     0    1228
>
1-sum(sabi.predrpart==sabi[ind,"ESTADO"])/length(sabi[ind,"ESTADO"])
[1] 0
```

Sin embargo, el error en el conjunto de prueba es del 13,86%. El comportamiento ante nuevas observaciones de este modelo aconseja un árbol más simple.

```
>  sabi.predrpart<-predict(sabi.rpart,  newdata  =  sabi[-ind,],
type="class")
> table(sabi.predrpart, sabi$ESTADO[-ind], dnn=c("Clase estimada",
"Clase real"))
              Clase real
Clase estimada Fracaso Activa
       Fracaso   114      15
       Activa     23     122
>1-sum(sabi.predrpart==sabi[-ind,"ESTADO"])/length(sabi[-ind,"ESTA
DO"])
[1] 0.1386861
```

La figura 9.2 recoge la evolución del error en validación cruzada conforme varía el tamaño del árbol. Este gráfico muestra como no se puede aumentar indefinidamente el

tamaño del árbol, ya que se estará sobreajustando. De esta forma, se debe detener el desarrollo del árbol una vez que el error de validación cruzada alcance el mínimo. Como se vio en el apartado 4.11.1, se aplica la regla 1-SE, según la cual se elige el árbol de menor tamaño cuyo error no supere un valor umbral. Este valor se calcula como el error mínimo más una vez su desviación típica y se representa en la figura 9.2 mediante la línea horizontal roja

```
> plotcp(sabi.rpart)
```

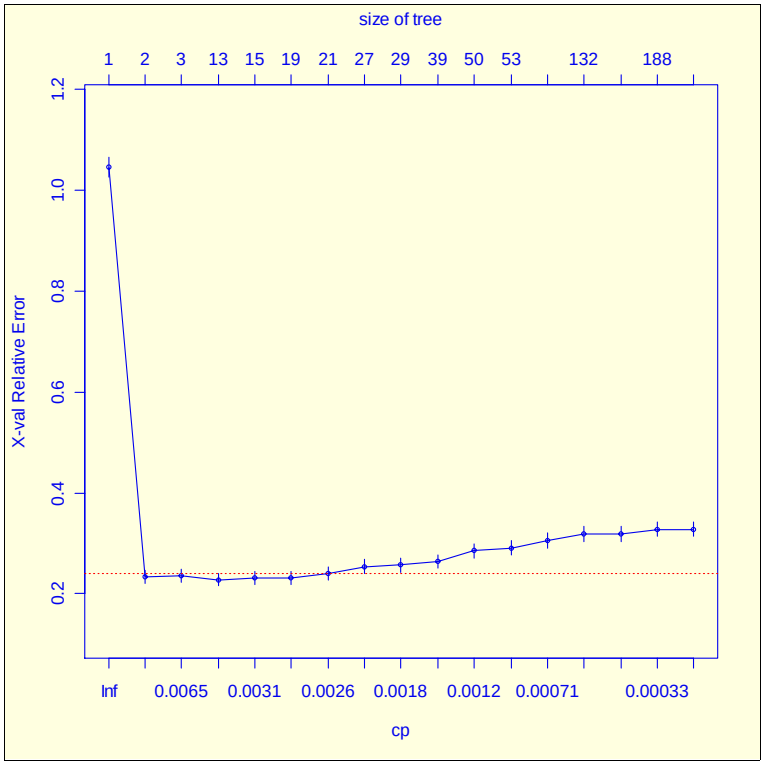


Figura 9.2 Evolución del error de validación cruzada al aumentar el tamaño del árbol, para el caso de dos clases.

El siguiente cuadro recoge la salida de la función rpart para el caso dicotómico. En primer lugar muestra la llamada que se ha hecho a la función y después las variables que realmente se han utilizado para desarrollar el árbol. Posteriormente, señala el error que existe en el nodo raíz, es decir, el error de la regla por defecto. Finalmente, la tabla recoge para cada corte el parámetro coste-complejidad, la posición que ocupa, el error

en el conjunto de entrenamiento o error relativo, el error de validación cruzada y la desviación típica del error de validación cruzada. Tanto el error del conjunto de entrenamiento como el de validación cruzada se obtienen multiplicando el valor que aparece en esa tabla por el error en el nodo raíz.

```
printcp(sabi.rpart)

Classification tree:
rpart(formula = ESTADO ~ ., data = sabi[ind, ], method = "class",
cp = 0, minsplit = 1, maxdepth = 30)

Variables actually used in tree construction:
 [1] AC.AT1   AC.PC1   BAI.AT1  BAI.FP1  CF.PT1   CNAE1    FM.AT1   FM.V1
 [9] IN.AT1   JURIDICA lnAC1    lnAT1    PE.PT1   T.AT1    T.PC1    V.AC1
[17] V.AT1    V.FP1

Root node error: 1228/2456 = 0.5
n= 2456

           CP nsplit  rel error  xerror     xstd
1  0.76710098      0 1.00000000 1.04560 0.020157
2  0.01221498      1 0.23289902 0.23453 0.012984
3  0.00346091      2 0.22068404 0.23534 0.013004
4  0.00325733     12 0.18566775 0.22801 0.012826
5  0.00298588     14 0.17915309 0.23127 0.012906
6  0.00285016     18 0.16693811 0.23046 0.012886
7  0.00244300     20 0.16123779 0.24023 0.013120
8  0.00203583     26 0.14657980 0.25407 0.013439
9  0.00162866     28 0.14250814 0.25651 0.013494
10 0.00122150     38 0.12622150 0.26384 0.013657
11 0.00108578     49 0.11074919 0.28502 0.014108
12 0.00081433     52 0.10749186 0.29072 0.014224
13 0.00061075    127 0.03175896 0.30537 0.014516
14 0.00054289    131 0.02931596 0.31840 0.014765
15 0.00040717    146 0.01872964 0.31840 0.014765
16 0.00027144    187 0.00081433 0.32818 0.014946
17 0.00000000    190 0.00000000 0.32818 0.014946
```

Como se ha dicho anteriormente este primer árbol está sobreajustado, por lo que consigue un ajuste perfecto en el conjunto de entrenamiento pero tiene un mal comportamiento a la hora de clasificar nuevas observaciones, por ejemplo, las del conjunto de test. Por ello, se realiza una poda de este árbol siguiendo la regla 1-SE, que, como ya se ha comentado, consiste en elegir el árbol de menor tamaño que tenga un error de validación cruzada igual o menor al mínimo más una desviación estándar (columnas xerror y xstd, de la tabla anterior, respectivamente).

```
> prune(sabi.rpart, cp=0.01)->sabi.prune
> printcp(sabi.prune)

Classification tree:
rpart(formula = ESTADO ~ ., data = sabi[ind, ], method = "class", cp
= 0, minsplit = 1, maxdepth = 30)

Variables actually used in tree construction:
[1] CF.PT1 PE.PT1

Root node error: 1228/2456 = 0.5

n= 2456

        CP nsplit rel error  xerror     xstd
1 0.767101      0   1.00000 1.04560 0.020157
2 0.012215      1   0.23290 0.23453 0.012984
3 0.010000      2   0.22068 0.23534 0.013004
```

Este árbol podado proporciona un error del 11,03% en el conjunto de entrenamiento, siendo su matriz de confusión la que se muestra a continuación.

```
>   sabi.predrpart<-predict(sabi.prune,   newdata   =   sabi[ind,],
type="class")
>1-sum(sabi.predrpart==sabi[ind,"ESTADO"])/length(sabi[ind, "ESTADO"])
[1] 0.1103420
>   table(sabi.predrpart,  sabi$ESTADO[ind],  dnn=c("Clase  estimada",
"Clase real"))
              Clase real
Clase estimada  Fracaso Activa
       Fracaso    1225     268
       Activa        3     960
```

Sin embargo, a pesar del aumento del error en el conjunto de entrenamiento ahora se comporta mejor en el conjunto de test, donde consigue un error del 9,124%. En la matriz de confusión para el conjunto de test se observan dos empresas fracasadas clasificadas como activas.

```
>   sabi.predrpart<-predict(sabi.prune,   newdata   =   sabi[-ind,],
type="class")
>1-sum(sabi.predrpart==sabi[-ind,"ESTADO"])/length(sabi[-ind,"ESTAD
O"])
[1] 0.09124088
>  table(sabi.predrpart, sabi$ESTADO[-ind], dnn=c("Clase estimada",
"Clase real"))
              Clase real
Clase estimada Fracaso Activa
       Fracaso 135      23
       Activa    2     114
```

El árbol podado presenta la siguiente estructura, que se puede ver de forma gráfica en la figura 9.3.

```
> sabi.prune
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1228 Fracaso (0.5000000 0.5000000)
  2) CF.PT1< 0.4915537 1514  286 Fracaso (0.8110964 0.1889036)
     4)PE.PT1< 2.536775 1493  268 Fracaso (0.8204956 0.1795044) *
     5) PE.PT1>=2.536775 21    3 Activa (0.1428571 0.8571429) *
  3) CF.PT1>=0.4915537 942    0 Activa (0.0000000 1.0000000) *
```

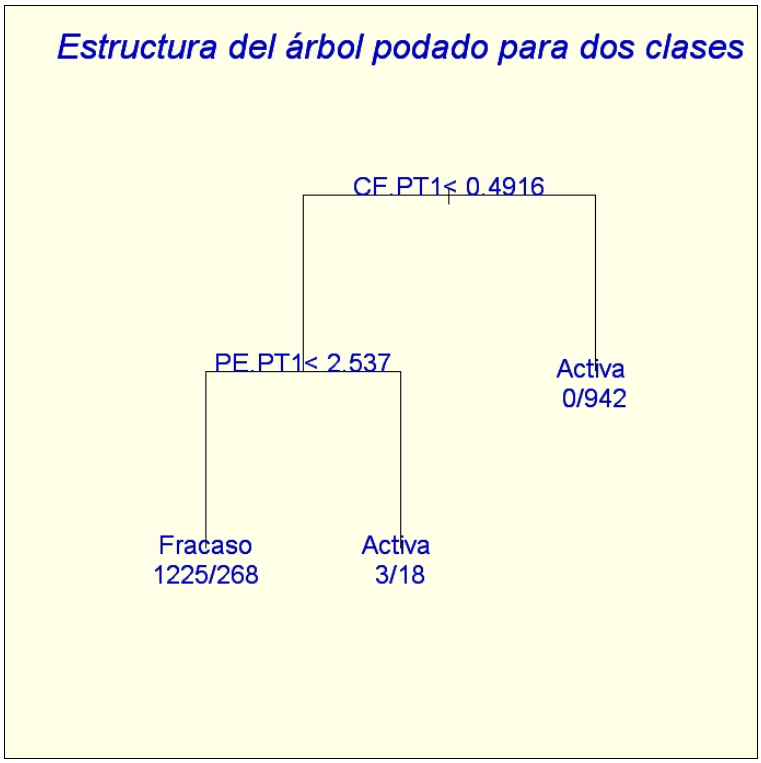


Figura 9.3 Estructura del árbol podado para el caso de dos clases.

En este caso el árbol podado únicamente utiliza los ratios CF.PT1 y PE.PT1 para discriminar entre las empresas activas y las fracasadas. Ambos ratios hacen referencia a la solvencia de la empresa.

9.4.2. Predicción del fracaso empresarial mediante Boosting. El caso de dos clases

Como se ha comentado previamente, a la hora de aplicar el método boosting se presentó el problema de que no estaba implementado en ningún programa estadístico, por lo que a la dificultad propia de este trabajo se une el hecho de haber tenido que aprender a programar en lenguaje R para implementar personalmente el algoritmo utilizado en boosting.

De los muchos algoritmos de la familia boosting comentados en el capítulo 7, se decidió implementar el algoritmo Adaboost.M1 debido a que es válido tanto para el caso dicotómico como para el caso general de *q* clases, por lo que es más adecuado para el objetivo de este trabajo. Este algoritmo fue propuesto por Freund y Schapire en 1996 en su artículo *"Experiments with a new boosting algorithm"* y supone una extensión natural del algoritmo Adaboost para dos clases, al caso general de más de dos clases.

Aunque el método boosting puede utilizar como clasificadores individuales cualquier tipo de clasificador, se optó por implementar el algoritmo utilizando árboles de clasificación como clasificadores básicos, entre otras cosas, porque es el que se utiliza en la mayoría de las aplicaciones de boosting, por los buenos resultados que proporcionan, y por la facilidad con que se manejan las variables cualitativas.

El único programa que se encontró con contenido parecido al deseado es una librería de R llamada LogitBoost (desarrollada por Marcel Dettling en 2003) que implementaba el algoritmo del mismo nombre descrito en el apartado 7.4 de este trabajo. Sin embargo, este conjunto de funciones no permitía la predicción de nuevas observaciones una vez entrenado el clasificador, por lo que se decidió no utilizarlo. Por tanto, se tuvo que implementar personalmente los algoritmos de adaboost.M1 y bagging en lenguaje R, planteándose como objetivo inicial la elaboración de al menos dos funciones para cada método, una que construyese el clasificador adaboost.M1 (o bagging) y clasificase a las observaciones del conjunto de entrenamiento, y otra que utilizase el clasificador

entrenado para predecir la clase de un conjunto de observaciones nuevas. Posteriormente, también se consideró interesante disponer de una función que permitiese aplicar validación cruzada sobre un conjunto de observaciones.

Cualquier función en R requiere de un conjunto de argumentos iniciales, en el caso de adaboost.M1, estos argumentos son los siguientes:

< La fórmula donde se especifica cuál es la variable dependiente y cuáles son las independientes.

< El nombre del objeto (data frame en la terminología de R) que contiene los datos que se quieren utilizar.

< Si se desea utilizar la técnica de remuestreo o la de reponderación.

< El número de iteraciones, o lo que es lo mismo el número de árboles a utilizar en la combinación.

< Y, por último, tres parámetros que son propios de la función *rpart* pero que se trasladan también a adaboost.M1 para limitar el tamaño del árbol.

La salida que devuelve esta función es, en terminología de R, una lista que contiene los siguientes aspectos:

< la fórmula utilizada,

< todos los árboles construidos,

< las ponderaciones correspondientes a los distintos árboles,

< la suma ponderada de los votos que las observaciones reciben para cada clase,

< la clase que se asigna a cada observación y,

< la importancia relativa de cada variable en la clasificación.

El método boosting utiliza un elevado número de árboles (cientos o tal vez miles). Esto dificulta la interpretación del clasificador final, y puede provocar que se considere este método como una caja negra. Para evitar esto es fundamental cualquier herramienta que ayude a comprender el funcionamiento del clasificador boosting. Por ello se ha programado la función adaboost.M1 para que proporcione una medida de la importancia de las características. La importancia relativa de las variables en el clasificador boosting se ha calculado a partir del número de veces que cada una de ellas es elegida para realizar un corte. En concreto, se suman las veces que cada característica ha sido elegida a lo largo de todos los árboles de las distintas iteraciones de boosting. Finalmente, se normaliza para que la suma de la importancia de todas las variables sea cien. Esta medida debe entenderse como una aproximación a la verdadera contribución de cada variable, que debería considerar la ganancia de información conseguida cada vez que se utiliza una variable para realizar un corte.

Para más información sobre las funciones desarrolladas veáse el Apéndice A. A continuación se recogen los resultados obtenidos al aplicar la función *adaboost.M1* sobre el conjunto de datos para el caso dicotómico, distinguiendo entre activas y fracasadas.

Una vez implementadas las funciones descritas, se construye un clasificador boosting con cien árboles desarrollados hasta $cp=0.01$, lo que limita el tamaño del árbol, pero de forma relativa según la complejidad de la muestra sobre la que se calcule el árbol en cada iteración del algoritmo boosting. En el apéndice B se pueden encontrar los árboles utilizados en la combinación. El error en el conjunto de test se reduce hasta el 6,569%, lo que supone una reducción del 28% comparado con el error del árbol podado que es de 9,124%. Además, si se analiza la matriz de confusión, puede verse como la mayoría de estos errores se cometen al clasificar como fracasadas empresas que continúan siendo activas. Los siguientes cuadros muestran, por este orden, el error en los conjuntos de entrenamiento y prueba.

```
>sabi.boosting <- adaboost.M1(ESTADO ~ ., data=sabi[ind,], mfinal=100,
boos=T,cp=0.01)
> table(sabi.boosting$clase, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
              Clase real
Clase estimada Fracaso Activa
       Fracaso 1204     196
       Activa    24    1032

>1-sum(sabi.boosting$clase==sabi$ESTADO[ind])/length(sabi$ESTADO[ind])
[1] 0.08957655
```

```
>predict.boosting(sabi.boosting, newdata=sabi[-ind,]) ->sabi.predboost
> sabi.predboost[-1]
$"confusión"
              Clase real
Clase estimada Fracaso Activa
       Fracaso 135      16
       Activa    2     121

$error
[1] 0.06569343
```

La función adaboost.M1 permite ver la importancia relativa de las variables, siendo en este caso los más destacados los ratios CF.PT1, PE.PT1 , BAI.FP1 y BAI.AT1 con valores en esta medida del 15'1, 12'5, 11'2, y 10'2% respectivamente. Destacan también, pero por su poca importancia, la variable categórica JURIDICA, y los ratios AC.PC1 y FM.V1 con valores inferiores al 2%. La tabla 9.1 recoge todas las variables ordenadas de mayor a menor importancia relativa.

Variable	Importancia relativa	Variable	Importancia relativa
CF.PT1	15,104167	lnAC1	3,906250
PE.PT1	12,500000	T.PC1	3,645833
BAI.FP1	11,197917	IN.AT1	3,385417
BAI.AT1	10,156250	V.AC1	2,864583
V.AT1	7,031250	lnAT1	2,604167
CNAE1	5,729167	FM.AT1	2,343750
V.FP1	5,729167	FM.V1	1,562500
T.AT1	4,687500	AC.PC1	1,562500
AC.AT1	4,687500	JURIDICA	1,302083

Tabla 9.1 Importancia relativa de las variables

La figura 9.4 recoge la matriz de gráficos de dispersión para las cuatro variables más importantes según esta medida. Las empresas fracasadas se representan en rojo y las activas en verde. La principal conclusión de esta matriz de gráficos es la dificultad para discriminar entre ambas clases de empresas a partir de dos variables. Esto muestra la dificultad del problema en cuestión y puede suponer un aviso de que la dificultad será aún mayor al considerar las tres clases.

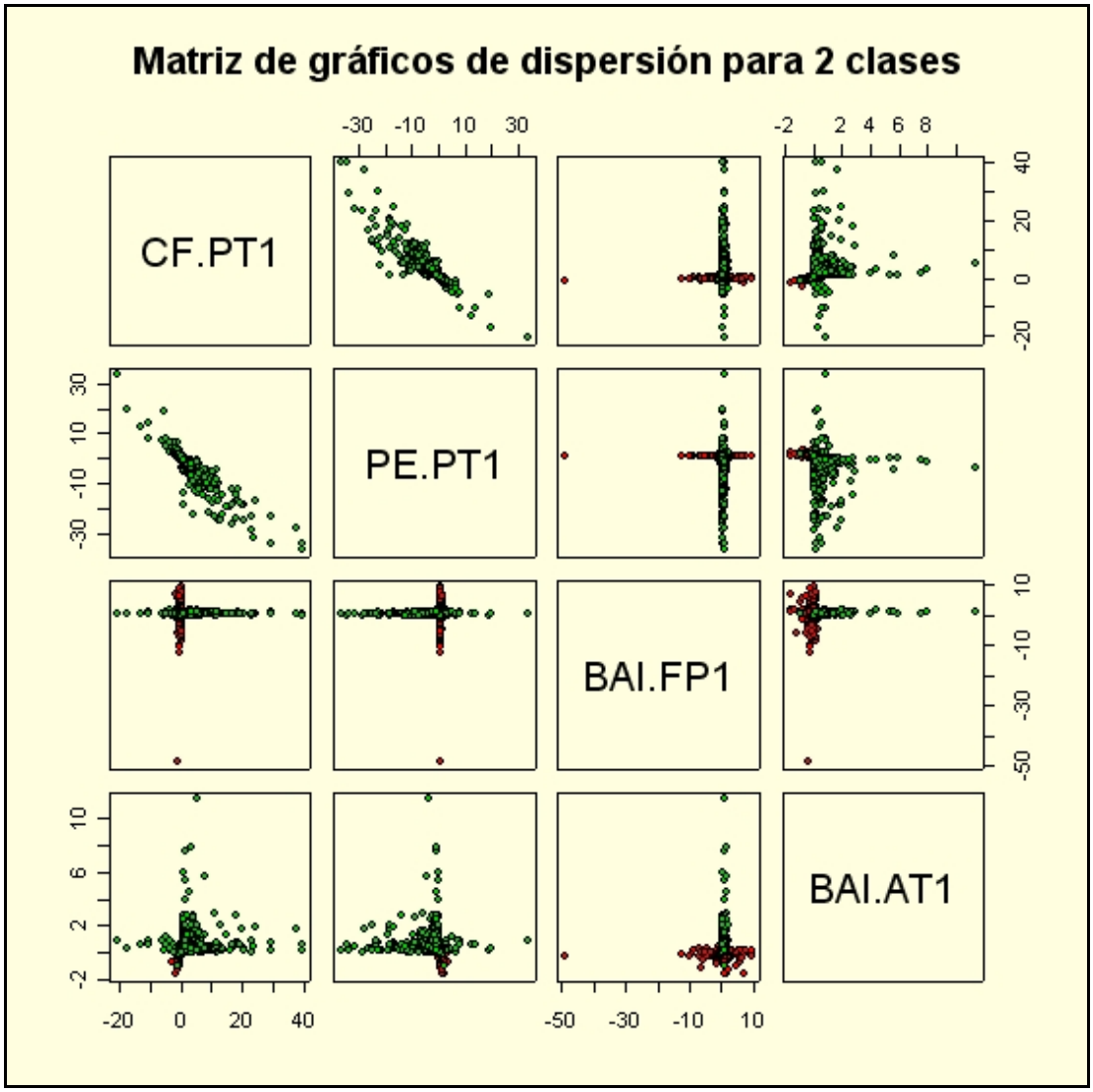


Figura 9.4 Matriz de gráficos de dispersión. Fracasadas en rojo y activas en verde

El siguiente gráfico muestra la distribución de los márgenes, tal y como se explicaron en el apartado 6.4.2 , para el algoritmo boosting desarrollado en este apartado. El eje *x* es el margen (*m*) y el eje *y* es el porcentaje de observaciones cuyo margen es menor o igual que *m*. Si todos los ejemplos de entrenamiento estuviesen clasificados correctamente, habría sólo margenes positivos. Además, los márgenes muestran el grado de seguridad en la predicción, de tal forma que si todas las observaciones estuviesen clasificadas correctamente y con la máxima certeza posible, el gráfico sería una línea vertical en *m*=1.

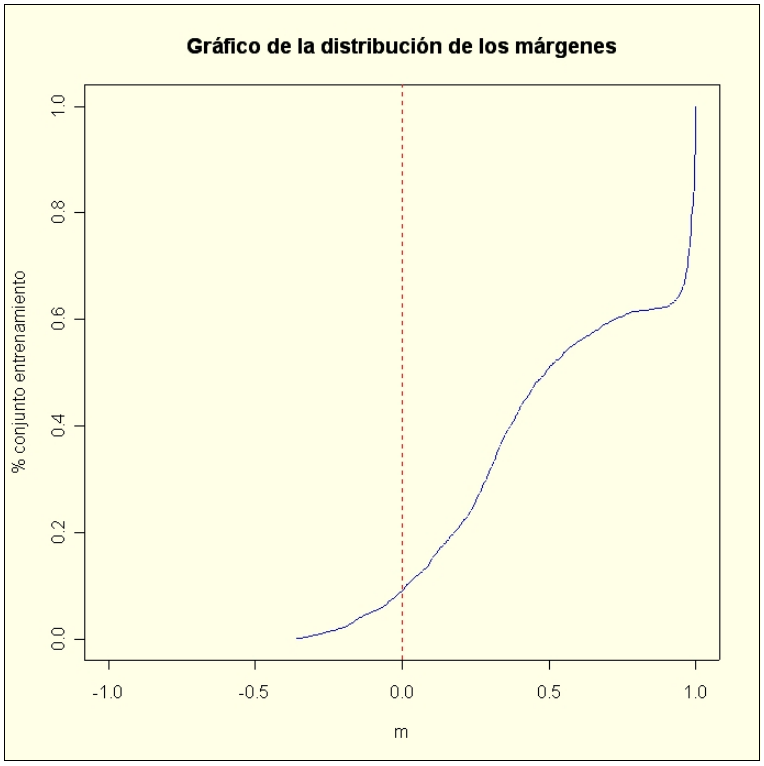


Figura 9.5 Distribución de los márgenes .

9.4.3. Predicción del fracaso empresarial con árboles de clasificación. El caso de tres clases

Lo más habitual en los trabajos realizados para la predicción del fracaso empresarial es considerar únicamente dos posibles estados, discriminando así entre empresas sanas (activas) y empresas fracasadas. En este apartado se intenta ver qué pasaría si se consideraran dos tipos de fracaso, fracaso1 y fracaso2. Dependiendo de si la empresa desaparece como tal organización, ya sea por disolución o por absorción (fracaso1), o si la empresa ha entrado en un proceso legal, quiebra o suspensión de pagos (fracaso 2).

La muestra con la que se trabaja tiene 512 observaciones de cada clase, de ellas el 90% se utiliza como conjunto de entrenamiento, y el resto para conjunto de test, manteniendo la proporcionalidad entre las clases. Por tanto, de las 1536 observaciones iniciales, 1383 se utilizan en el conjunto de entrenamiento para construir el clasificador

y, el resto, 153 se mantienen ocultas para el clasificador y se utilizan como conjunto de prueba.

En primer lugar se desarrolla el árbol completamente, para ello se fija el parámetro de coste-complejidad en 0 (cp=0) y el minsplit=2.

```
> sabi.rpart<-rpart(ESTADO~.,method="class", data=sabi[ind,], cp=0,
minsplit=2, maxdepth=30)

>     sabi.predrpart<-predict(sabi.rpart,   newdata   =   sabi[ind,],
type="class")
>    1-sum(sabi.predrpart==sabi[ind,"ESTADO"])/length(sabi[ind,
"ESTADO"])
[1] 0
>  table(sabi.predrpart, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
                          Clase real
Clase estimada    Activa      Fracaso1    Fracaso2
Activa            461         0           0
Fracaso1          0           461         0
Fracaso2          0           0           461
```

De esta forma, se consigue un árbol que ajusta perfectamente el conjunto de entrenamiento, como se puede ver en la anterior matriz de confusión. Sin embargo, si se aplica sobre el conjunto de prueba o test, los resultados son menos favorables cometiendo un error de generalización del 30,72%. Además, hay 14 empresas fracasadas clasificadas como activas (8 de fracaso1 y 6 de fracaso2). La figura 9.6 muestra la evolución del error de validación cruzada a medida que aumenta la complejidad del árbol.

```
>     sabi.predrpart<-predict(sabi.rpart,   newdata   =   sabi[-ind,],
type="class")
>   table(sabi.predrpart, sabi$ESTADO[-ind], dnn=c("Clase estimada",
"Clase real"))
                              Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            43          8           6
Fracaso1          4           29          11
Fracaso2          4           14          34
>1-sum(sabi.predrpart==sabi[-ind,"ESTADO"])/length(sabi[-ind,
"ESTADO"])
[1] 0.3071895
```

```
> plotcp(sabi.rpart)
```

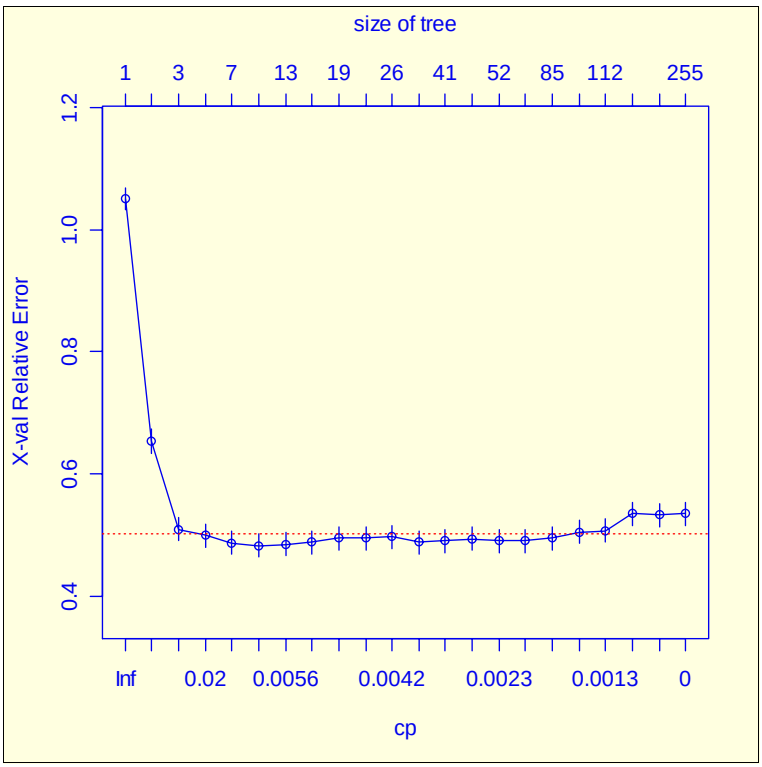


Figura 9.6 Evolución del error de validación cruzada conforme aumenta el tamaño del árbol, el caso de tres clases.

```
> printcp(sabi.rpart)
Classification tree:
rpart(formula = ESTADO ~ ., data = sabi[ind, ], method = "class",
    cp = 0, minsplit = 1, maxdepth = 30)

Variables actually used in tree construction:
 [1] AC.AT1   AC.PC1   BAI.AT1  BAI.FP1  CF.PT1   CNAE1    FM.AT1
FM.V1
 [9] IN.AT1   JURIDICA  lnAC1    lnAT1    PE.PT1   T.AT1    T.PC1
V.AC1   V.AT1    V.FP1

Root node error: 922/1383 = 0.66667
n= 1383
      CP        nsplit rel error  xerror  xstd
1  0.37418655      0  1.000000 1.04989 0.018485
2  0.12798265      1  0.625813 0.65401 0.020002
3  0.03687636      2  0.497831 0.51085 0.019115
4  0.01084599      3  0.460954 0.50000 0.019014
5  0.00650759      6  0.426247 0.48807 0.018898
6  0.00585683      7  0.419740 0.48373 0.018854
7  0.00542299     12  0.390456 0.48590 0.018876
8  0.00498915     13  0.385033 0.48915 0.018908
9  0.00488069     18  0.360087 0.49566 0.018972
10 0.00433839     20  0.350325 0.49566 0.018972
11 0.00397686     25  0.328633 0.49783 0.018993
12 0.00325380     28  0.316703 0.48915 0.018908
13 0.00271150     40  0.277657 0.49132 0.018930
14 0.00253073     42  0.272234 0.49458 0.018962
15 0.00216920     51  0.248373 0.49132 0.018930
16 0.00180766     78  0.185466 0.49132 0.018930
17 0.00162690     84  0.174620 0.49566 0.018972
18 0.00144613    105  0.137744 0.50651 0.019075
19 0.00108460    111  0.129067 0.50868 0.019095
20 0.00072307    203  0.029284 0.53579 0.019327
21 0.00054230    212  0.022777 0.53362 0.019310
22 0.00000000    254  0.000000 0.53579 0.019327
```

Como se ha dicho anteriormente, este primer árbol está sobreajustado, por lo que consigue un ajuste perfecto en el conjunto de entrenamiento, pero tiene un mal comportamiento a la hora de clasificar nuevas observaciones, por ejemplo, las del conjunto de test. Por ello, se realiza una poda de este árbol siguiendo la regla 1-SE que, como se vio en el apartado 4.11.1, consiste en elegir el árbol de menor tamaño que tenga un error de validación cruzada igual o menor al mínimo más una desviación estándar (columnas xerror y xstd, de la tabla anterior, respectivamente).

```
> sabi.prune<-prune(sabi.rpart,cp=0.03)
> printcp(sabi.prune)

Classification tree:
rpart(formula = ESTADO ~ ., data = sabi[ind, ], method = "class",
    cp = 0, minsplit = 1, maxdepth = 30)

Variables actually used in tree construction:
[1] lnAT1  PE.PT1

Root node error: 922/1383 = 0.66667

n= 1383

        CP nsplit rel error  xerror     xstd
1 0.374187      0   1.00000 1.05748 0.018395
2 0.127983      1   0.62581 0.67245 0.020059
3 0.036876      2   0.49783 0.51302 0.019134
4 0.030000      3   0.46095 0.50217 0.019035
```

Se calcula la matriz de confusión y el error en el conjunto de entrenamiento

```
>     sabi.predrpart<-predict(sabi.prune,  newdata  =  sabi[ind,],
type="class")
>   table(sabi.predrpart, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
                          Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            345         0           0
Fracaso1          83          284         132
Fracaso2          33          177         329
>1-sum(sabi.predrpart==sabi[ind,"ESTADO"])/length(sabi[ind,"ESTADO"])
[1] 0.3073030
```

Se calcula la matriz de confusión y el error en el conjunto de test

```
>   sabi.predrpart<-predict(sabi.prune,   newdata   =   sabi[-ind,],
type="class")
>  table(sabi.predrpart, sabi$ESTADO[-ind], dnn=c("Clase estimada",
"Clase real"))
                         Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            42          0           0
Fracaso1          6           33          13
Fracaso2          3           18          38

>1-sum(sabi.predrpart==sabi[-ind,"ESTADO"])/length(sabi[-ind,
"ESTADO"])
[1] 0.2614379
```

Como se puede ver, se ha reducido el error en el conjunto de test del 30,72%, con el árbol completamente desarrollado, a un 26,144% con el árbol podado y, lo que es quizá más importante, se ha conseguido solventar el problema de las empresas pertenecientes al grupo de fracaso1 y fracaso2 que eran clasificadas como activas, 8 y 6

respectivamente. Se comprueba así la conveniencia de podar los árboles, ya que, a pesar del incremento del error en el conjunto de entrenamiento, lo importante es el error de generalización que se estima de forma más precisa en el conjunto de prueba. El árbol podado tiene la siguiente estructura, que se puede representar en la figura 9.6. Las variables que realmente se utilizan son el ratio PE.PT1, que ya se utilizaba para el árbol podado cuando sólo había dos clases y que es un ratio de solvencia, y el lnAT1 que, como ya se ha dicho, es una variable indicativa del tamaño de la empresa.

```
> sabi.prune
n= 1383
node), split, n, loss, yval, (yprob)
      * denotes terminal node
 1) root 1383 922 Fracaso1 (0.33333333 0.33333333 0.33333333)
   2) PE.PT1>=-0.01480628 1038 577 Fracaso1 (0.44412331 0.11175337 0.44412331)
     4) PE.PT1< 0.6822301 311 126 Fracaso1 (0.59485531 0.18971061 0.21543408) *
     5) PE.PT1>=0.6822301 727 333 Fracaso2 (0.37964237 0.07840440 0.54195323)
      10) lnAT1< 6.546879 188  89 Fracaso1 (0.52659574 0.12765957 0.34574468) *
      11) lnAT1>=6.546879 539 210 Fracaso2 (0.32838590 0.06122449 0.61038961) *
   3) PE.PT1< -0.01480628 345   0 Activa (0.00000000 1.00000000 0.00000000) *
```

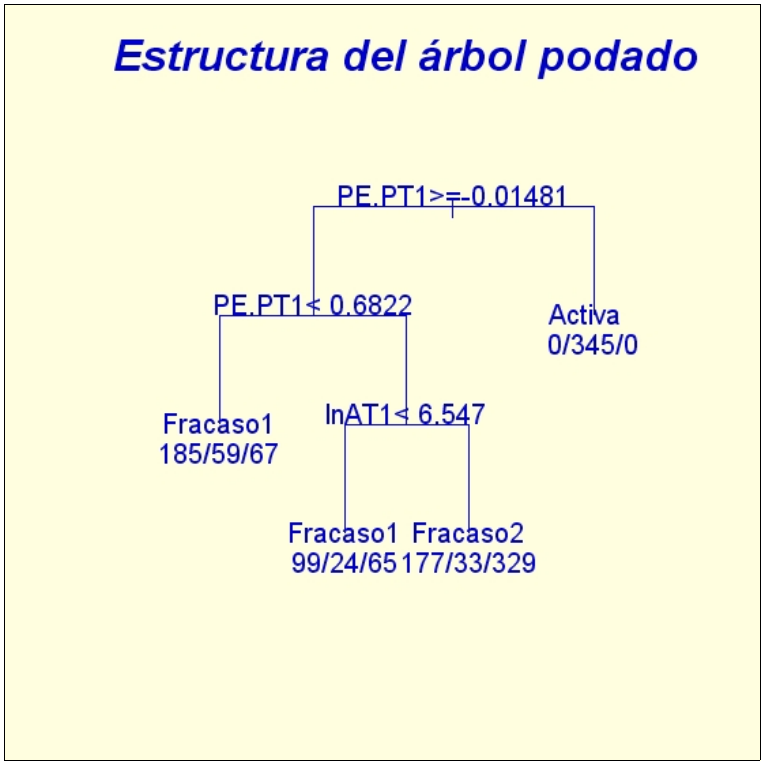


Figura 9.7 Estructura del árbol podado para tres clases.

9.4.4. Predicción del fracaso empresarial mediante Boosting. El caso de tres clases

En primer lugar se aplica boosting fijando en 100 el número de iteraciones y permitiendo un desarrollo del árbol considerable, para lo que se establece el parámetro cp=0.005, frente al valor 0.03 que se establecía en la sección anterior para podar el árbol. A continuación se muestran los errores en el conjunto de entrenamiento y en el conjunto de prueba o test.

```
> sabi.boosting <- adaboost.M1(ESTADO ~ ., data=sabi[ind,],
mfinal=100, boos=T,maxdepth=30, cp=0.005)
>table(sabi.boosting$clase, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
                          Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            461         0           0
Fracaso1          0           461         0
Fracaso2          0           0           461
>1-sum(sabi.boosting$clase==sabi$ESTADO[ind])/length(sabi$ESTADO[ind])
[1] 0
```

```
>predict.boosting(sabi.boosting,    newdata=sabi[-ind,])    ->
sabi.predboost
>   sabi.predboost[-1]
$"confusión"
                            Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            42          1           1
Fracaso1          5           36          12
Fracaso2          4           14          38

$error
[1] 0.2418301
```

Se consigue un ajuste perfecto en el conjunto de entrenamiento y reducir el error en el conjunto de prueba hasta el 24,183%, lo que comparado con el 26,144% del árbol podado individual supone una reducción del 7,5%, y resulta aún más llamativo si se compara con el error del primer árbol construido (30,72%) pues supone una reducción del 21,28%. Además, esto se ha conseguido sin incrementar demasiado el número de empresas fracasadas clasificadas como activas, sólo una empresa del conjunto de test en cada tipo de fracaso.

La función *adaboost.M1* permite ver la importancia relativa de las variables, según el número de veces que se han utilizado cada una a lo largo de las distintas iteraciones de boosting (en este caso 100). En este caso ninguna variable tiene un papel predominante, no superando ninguna de ellas una importancia relativa del 10%. Si bien el mayor valor es para el BAI.AT1 con un 8,9% y la variable JURIDICA es la que menos interviene con sólo un 1,315%.

```
> sabi.boosting$import
JURIDICA    CNAE1   IN.AT1  BAI.AT1  BAI.FP1   AC.AT1    T.AT1    V.FP1
1.315257 7.830838 5.078915 8.903278 6.029947 7.405909 6.070417 6.212060
  AC.PC1    T.PC1    FM.V1   PE.PT1   CF.PT1   FM.AT1    V.AT1    V.AC1
4.593282 5.382436 3.358964 7.324970 7.264265 3.116147 1.881829 4.937272
   lnAT1    lnAC1
7.244031 6.050182
```

El siguiente gráfico muestra la distribución de los márgenes. En este caso, como el ajuste en el conjunto de entrenamiento es perfecto no hay ningún ejemplo con margen negativo. Además, aproximadamente un 30% de las observaciones tienen márgenes superiores a 0'5, lo que demuestra una elevada seguridad en su predicción.

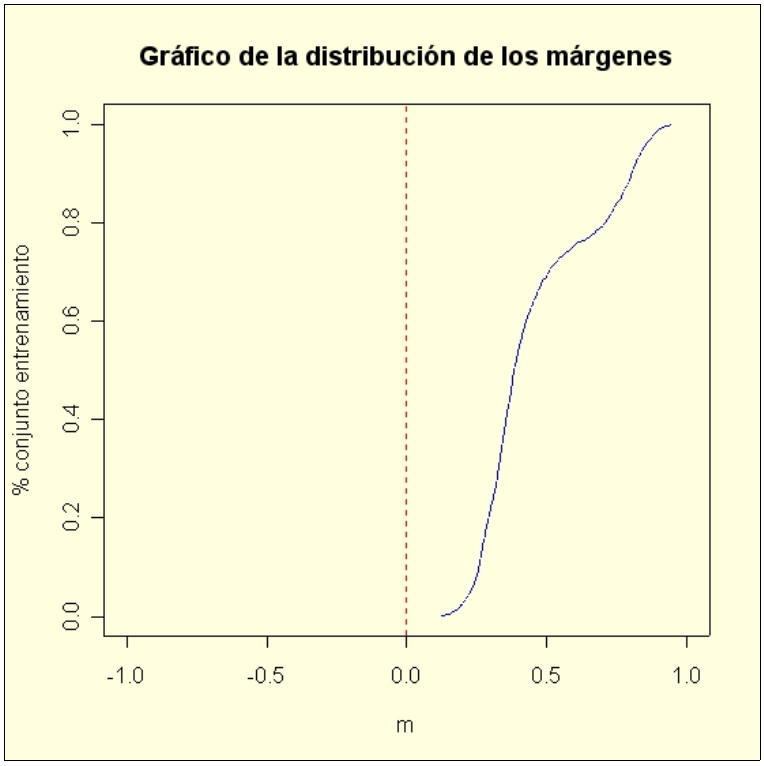


Figura 9.8 Distribución de los márgenes

Para ver el comportamiento de boosting ante la complejidad de los árboles, cuestión a la que se hacía referencia en el apartado 7.6 de este trabajo, se utilizará ahora el método boosting pero con árboles menos desarrollados para lo que se limita el valor de máxima profundidad del árbol (maxdepth=3), por lo que habrá que utilizar un mayor número de iteraciones. En primer lugar se muestra el error de entrenamiento y posteriormente el de prueba.

```
> sabi.boosting1 <- adaboost(ESTADO ~ ., data=sabi[ind,], mfinal=1000,
boos=T,maxdepth=3, cp=0.005)
>table(sabi.boosting1$clase, sabi$ESTADO[ind], dnn=c("Clase estimada",
"Clase real"))
                          Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            365         3           0
Fracaso1          67          310         94
Fracaso2          29          148         367

>1-sum(sabi.boosting1$clase==sabi$ESTADO[ind])/length(sabi$ESTADO[i
nd])
[1] 0.2465654
```

```
>   predict.boosting(   sabi.boosting1,   newdata=sabi[-ind,])   ->
sabi.predboost1

>   sabi.predboost[-1]
$"confusión"
                          Clase real
clase estimada    Activa      Fracaso1    Fracaso2
Activa            42          0           0
Fracaso1          5           37          10
Fracaso2          4           14          41

$error
[1] 0.2156863
```

A pesar de que el error en el conjunto de entrenamiento ha aumentado, se consigue reducir el error en el conjunto de prueba hasta el 21,569%, lo que comparado con el 26,144% del árbol podado individual supone una reducción del 17,5%, y resulta aún más llamativo si se compara con el error del primer árbol construido (30,72%), pues supone una reducción del 29,79%. Además, esto se ha conseguido manteniendo el número de empresas fracasadas clasificadas como activas a cero en ambos tipo de fracaso.

Si se analiza la matriz de confusión anterior desde la perspectiva dicotómica más habitual en los estudios sobre fracaso empresarial, empresas activas frente a empresas fracasadas, agrupando los dos tipos de fracaso, el error sería de 9/153= 0,0588 (5,88%).

	Clase real	
Clase estimada	**Activa**	**Fracaso**
Activa	42	0
Fracaso	9	102

Tabla 9.2 Matriz de confusión agrupando fracaso1 y fracaso2

La función adaboost.M1 permite ver la importancia relativa de las variables, según el número de veces que se ha utilizado cada una a lo largo de las distintas iteraciones de boosting (en este caso 1000). Ahora sí que hay más diferencias entre los distintos ratios, mostrándose como más importante el ratio PE.PT1 (23,61%), seguido de los ratios BAI.AT1 (14,92%) y CF.PT1 (11,07%). A continuación, y a cierta distancia, las variables CNAE1, T.PC1, V.AT1, lnAT1 y lnAC1. Estas variables toman valores que oscilan entre el 7,19% del lnAC1 y el 5,45% del lnAT1. El resto de variables quedan ya muy por detrás, siendo las de menor importancia según este indicador FM.AT1, FM.V1 y IN.AT1. La tabla 9.3 muestra, de mayor a menor, todas las variables ordenadas según su importancia relativa.

Por tanto, tanto en el caso dicotómico como en el de tres clases, los ratios de PE.PT1 y CF.PT1 están entre los tres más importantes, siendo estos ratios indicativos de la solvencia de la empresa. Además, los ratios de BAI.FP1 y BAI.AT1, para el caso de dos y tres clases, respectivamente, también están entre los tres más destacados. Estos dos ratios son indicativos de rentabilidad, ya sea financiera, en el primer caso, o económica, en el segundo. Estos resultados son coherentes con los de estudios previos de predicción de fracaso empresarial comentados en el apartado 8.2.3 de este trabajo.

Variable	Importancia relativa	Variable	Importancia relativa
PE.PT1	23,6096257	V.AC1	2,7540107
BAI.AT1	14,9197861	AC.AT1	2,5133690
CF.PT1	11,0695187	T.AT1	1,9786096
lnAC1	7,1925134	V.FP1	1,7647059
CNAE1	6,7914439	AC.PC1	1,6577540
V.AT1	6,4438503	JURIDICA	1,6042781
T.PC1	6,1497326	IN.AT1	0,8288770
lnAT1	5,4010695	FM.V1	0,4812834
BAI.FP1	4,4919786	FM.AT1	0,3475936

Tabla 9.3 Importancia relativa de las variables

La figura 9.9 recoge la matriz de gráficos de dispersión para las tres variables más importantes cuando se consideran las tres clases. Las empresas de la clase fracaso1 se representan en rojo, las activas en verde y las de la clase fracaso2 en azul. Se puede ver la dificultad para discriminar entre las tres clases de empresas en base a dos variables. Esto muestra la complejidad del problema en cuestión, lo que explica los niveles de error elevados.

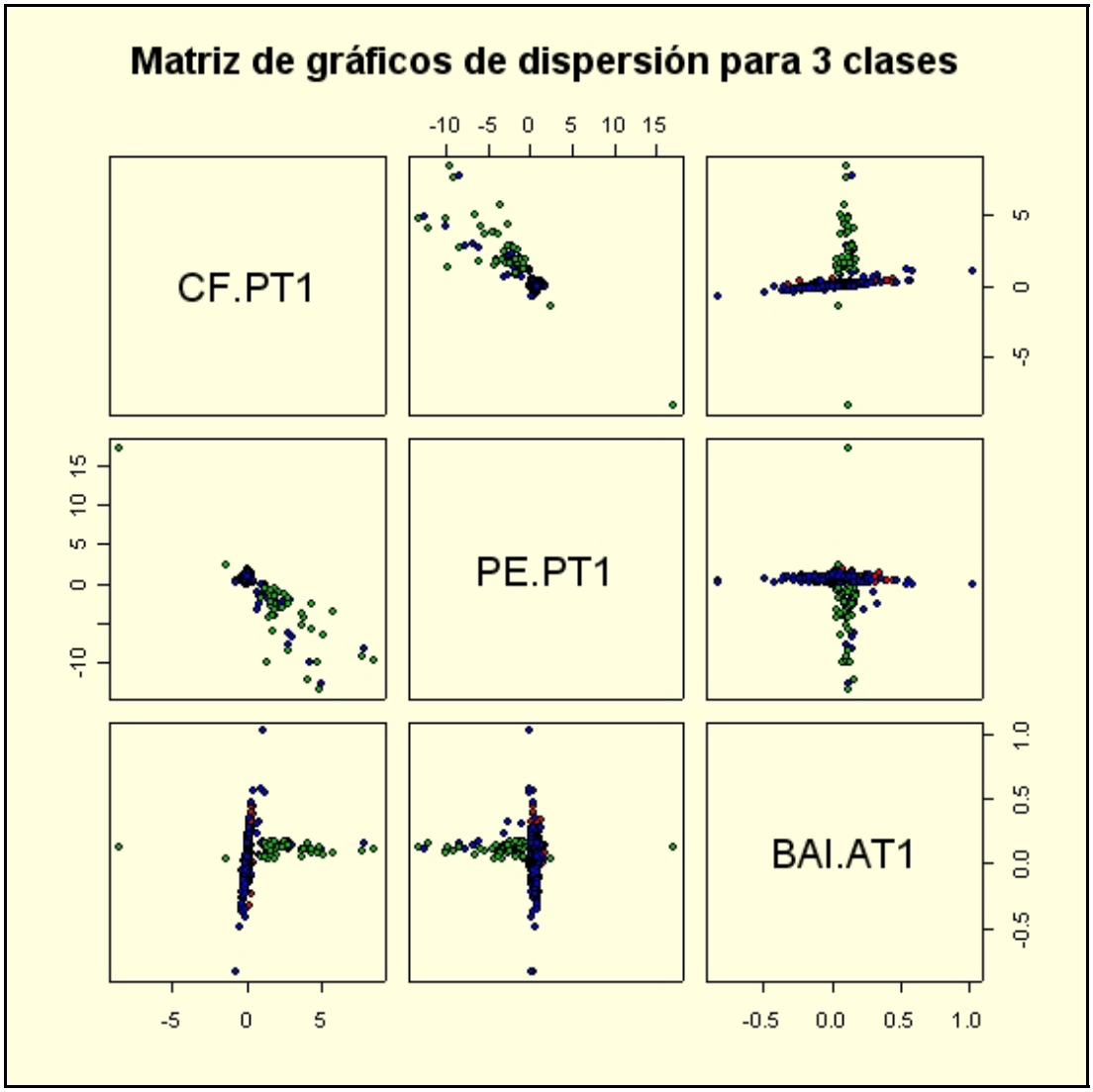


Figura 9.9 Matriz de gráficos de dispersión. Fracaso1 en rojo, activas en verde y fracaso2 en azul

La figura 9.10 muestra la distribución de los márgenes en el conjunto de entrenamiento, para el algoritmo boosting con 1000 iteraciones desarrollado en este apartado. Como ya se ha explicado el eje *x* es el margen (*m*) y el eje *y* es el porcentaje de observaciones cuyo margen es menor o igual que *m*. Si todos los ejemplos de entrenamiento estuviesen clasificados correctamente, habría sólo margenes positivos. En este caso hay un 24,65% de márgenes negativos que se corresponde con el error en el conjunto de test. También hay que destacar, que entorno al 20% de las observaciones tienen márgenes cercanos a la unidad (indica las empresas clasificadas con probabilidad uno).

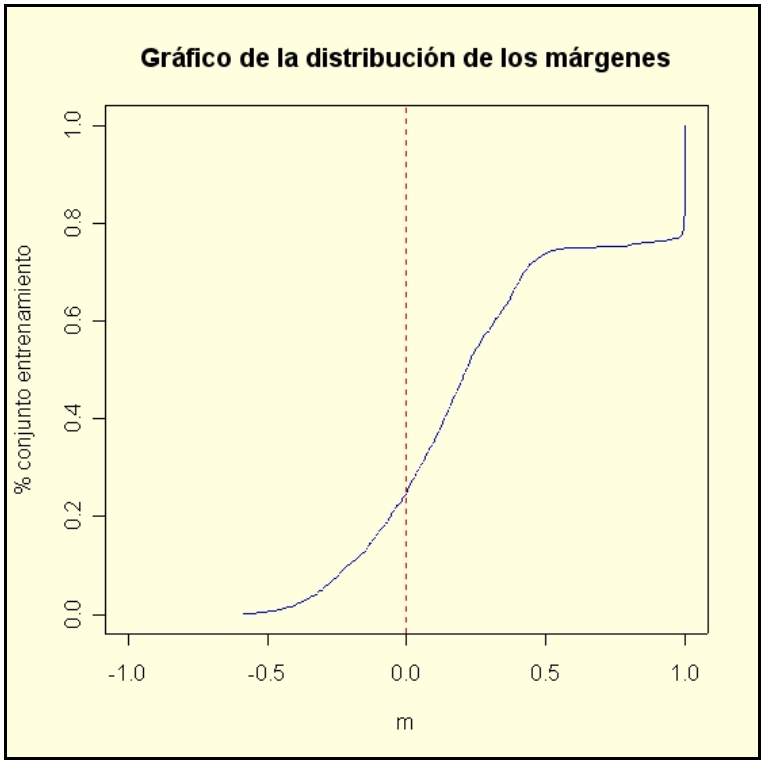


Figura 9.10 Distribución de los márgenes

9.4.5. Comparación de boosting con otros métodos

En este apartado se aborda la tarea de comparar boosting con algunos de los métodos de clasificación más utilizados. En realidad es muy difícil llevar a cabo una comparación exhaustiva, debido a que el abanico de posibilidades es muy amplio. Hay que seleccionar qué métodos de combinación se van a estudiar y qué clasificadores básicos se van a utilizar, así como el número de éstos, porque no es lo mismo, por ejemplo, comparar boosting y bagging aplicados sobre análisis discriminante lineal, que sobre árboles de clasificación o redes neuronales. Por todo esto se ha creído conveniente recoger en esta sección algunas de las comparaciones más interesantes que se han realizado, confrontando el comportamiento de boosting con el de otros métodos, antes de presentar la comparación que se ha realizado utilizando el conjunto de datos de este trabajo.

a) Comparaciones más destacadas en la literatura

A pesar del enorme interés que han suscitado los métodos de combinación de clasificadores en la última década y de los muchos trabajos realizados sobre este tema, existen pocas comparaciones entre los distintos métodos. Entre los estudios más interesantes, está el realizado por Breiman (1996) en este caso se comparan los métodos de Bagging y Boosting (Adaboost) en cinco conjuntos disponibles en la base datos de la Universidad de California, Irvine (UCI repository). Como clasificadores base se utilizan árboles de clasificación construidos mediante el sistema CART. Se comparan los porcentajes de error en el conjunto de prueba de estos métodos para un número de iteraciones determinado a priori (B=50) y para el caso en que se detiene el procedimiento cuando se alcanza un error de entrenamiento nulo.

Conjuntos de datos	Boosting		Bagging	
	B=50	error=0	B=50	error=0
heart	1,1	5,3	2,8	3,0
breast cancer	3,2	4,9	3,7	4,1
ionosphere	6,4	9,1	7,9	9,2
diabetes	26,6	28,6	23,9	24,7
glass	22,0	28,1	23,2	25

Tabla 9.4 Fuente: Breiman (1996)

Estos resultados confirman que boosting reduce el error de entrenamiento con mayor rapidez que bagging, pero que esa reducción no se produce tan rápidamente en el error de prueba. Además, se puede ver como boosting logra una mayor reducción del error para un mismo número de iteraciones, salvo para el problema de diabetes.

En segundo lugar, cabe destacar la evaluación empírica de bagging y boosting que realizan Maclin y Opitz (1997) utilizando 23 conjuntos de la base de datos de la universidad de California mencionada anteriormente. Este estudio tiene la virtud de comparar estos dos métodos utilizando como clasificadores base árboles de clasificación (CART) y redes neuronales. En el caso de las redes neuronales, también compara una combinación de varios clasificadores obtenidos a partir de valores iniciales aleatorios de los pesos. La tasa de aprendizaje se fija en 0,15 y el término impulso en 0,9. El

número de nodos ocultos se elige en función del número de unidades de entrada y salida de cada problema, y el número de iteraciones se basa en el número de ejemplos y en la estructura de la red.

Conjuntos de datos		Red Neuronal				CART		
		RN	Pesos aleat	bagging	boosting	CART	bagging	boosting
1	breast cancer	3,3	3,4	3,3	3,9	5	3,3	3,1
2	credit-a	14,8	14	14,1	16,2	14,9	12,1	12,6
3	credit-g	28,3	24,4	24,3	26,4	29,6	22,8	22,9
4	diabetes	23,6	22,8	23,2	22,8	28,3	21,9	22,3
5	glass	38,5	35,5	33,7	33,2	30,9	28,4	30,5
6	heart-cleveland	18,2	17,3	16,7	19,1	24,3	18,1	17,4
7	hepatitis	19,9	19,6	18,1	18,1	21,6	16,5	13,8
8	house-votes-84	5	4,9	4,3	5,1	3,5	3,6	4,4
9	hypo	6,4	6,2	6,2	6,2	0,5	0,4	0,4
10	ionosphere	10,1	8	7,6	8	8,1	6	6
11	iris	4,3	4	4,3	3,3	6	4,6	5,6
12	kr-vs-kp	2,3	0,9	0,9	0,4	0,6	0,5	0,3
13	labor	5,3	4,2	4,9	5	15,1	13,3	13,2
14	letter	18	12,8	12,5	4,6	14	10,6	6,7
15	promoters-936	5	4,8	4,5	4,7	12,8	9,5	6,3
16	ribosome-blind	9,5	8,5	8,4	8,5	11,2	9,3	9,1
17	satellite	12,9	11,1	11	10,3	13,8	10,8	10,4
18	segmentation	6,7	5,6	5,3	3,7	3,7	2,8	2,3
19	sick	6	5,7	5,8	4,8	1,3	1	0,9
20	sonar	16,9	16,7	16,5	12,5	29	21,6	19,7
21	soybean	9	6,4	6,8	6,3	8	8	7,9
22	splice	4,7	4	3,9	4,3	5,9	5,7	6,3
23	vehicle	24,5	21,1	21,7	19,5	29,4	26,1	24,8

Tabla 9.5 Fuente Maclin y Opitz (1997)

Para cada uno de los 23 conjuntos utilizados se ha señalado la casilla del método con menor tasa de error en el conjunto de prueba, los métodos que resultan vencedores en más ocasiones son boosting aplicado tanto a redes neuronales como a árboles de clasificación, y bagging aplicado a CART. Solamente en una ocasión resulta ganador uno de los sistemas individuales, luego se constata la superioridad de los métodos de combinación sobre los métodos de clasificación individuales. Ahora bien, la elección

de un método de combinación no parece sencilla a la luz de estos resultados, ya que ningún método mantiene una supremacía de forma general sobre el resto, como se puede ver en la figura 9.11.

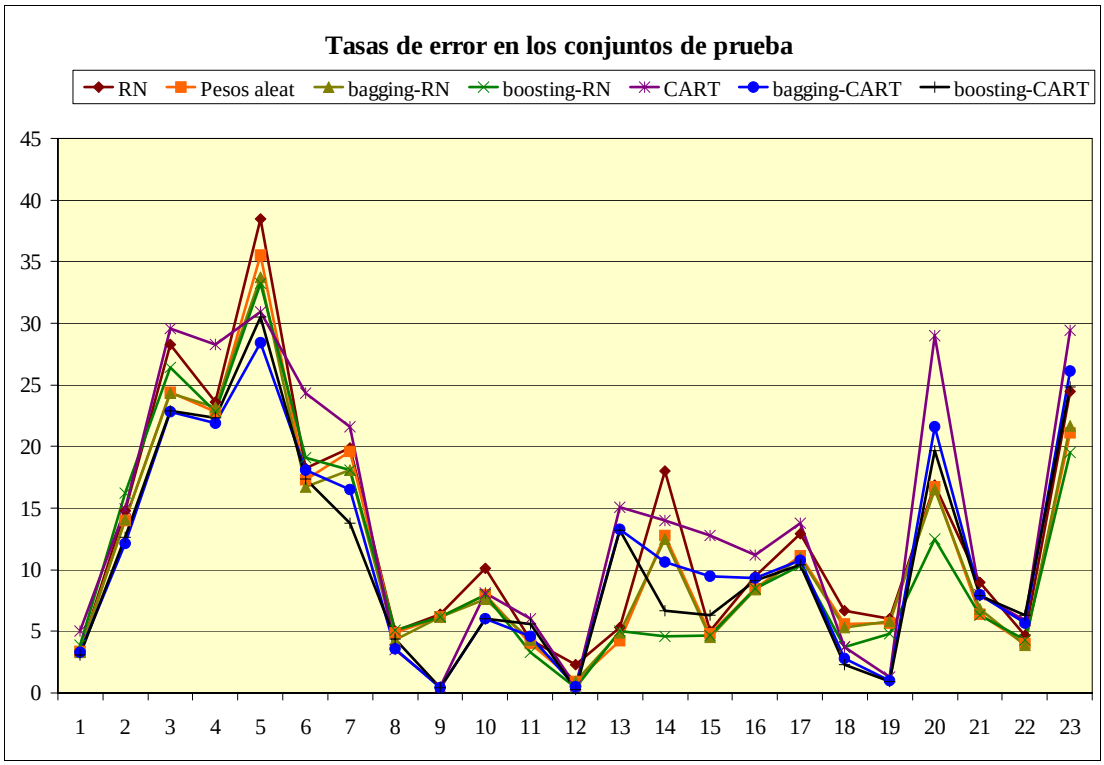


Figura 9.11 Tasas de error en los conjuntos de prueba

b) Comparación con los datos de fracaso empresarial

Una vez expuestas dos de las comparaciones más importantes entre el método boosting y otros sistemas de clasificación, se describe a continuación la comparación realizada utilizando el conjunto de datos para la predicción del fracaso empresarial con información del último año anterior al fracaso. Continuando con el mismo esquema utilizado anteriormente se plantea primero el problema en términos dicotómicos, distinguiendo entre empresas fracasadas y activas y, posteriormente, se amplia el problema al caso de tres clases considerando dos niveles de fracaso, fracaso1 y fracaso2.

En la comparación se van a utilizar siete clasificadores: el vecino más próximo, discriminante lineal, árbol de clasificación, bosque aleatorio, redes neuronales, bagging y boosting. Excepto bagging y boosting, para todos los demás existen librerías en el programa R que permiten su aplicación. En concreto, para construir los árboles se ha utilizado la librería *rpart*, para el discriminante lineal la librería *MASS*, para el vecino más próximo la librería *class*, para el bosque aleatorio la librería *randomForest* y, por último, en el caso de las redes neuronales se ha utilizado la librería llamada *nnet*. Cada una de estas funciones tiene una serie de parámetros a ajustar y cuyo significado se puede conocer a través de la ayuda que posee cada librería.

Cuando ha sido posible, se ha trabajado con los valores que las funciones atribuyen por defecto, para evitar la arbitrariedad en la elección de los parámetros. En el caso del vecino más próximo se ha utilizado k=5, considerando por tanto las cinco observaciones más próximas a la que se desea clasificar. Para el bosque aleatorio se han utilizado 100 iteraciones en cada repetición y para las redes neuronales se ha establecido 10 neuronas en la capa oculta manteniendo en el resto de parámetros los valores fijados por defecto en la función.

Metodo clasificación	2 clases (1)		3 clases (2)		3 clases (3)	
	error	desv tip	error	desv tip	error	desv tip
Vec Prox	0,1307	0,0168	0,3405	0,0245	0,3388	0,0229
Disc. Lineal	0,1256	0,0199	0,3235	0,0253	0,3308	0,0261
CART	0,1118	0,0164	0,3079	0,0250	0,3088	0,0238
R. Forest	0,1067	0,0176	0,2755	0,0211	0,2832	0,0252
Red Neur	0,1158	0,0120	0,3178	0,0277	0,3225	0,0280
Bagging	0,1129	0,0171	0,2986	0,0278	0,3021	0,0231
Boosting	0,1045	0,0153	0,2786	0,0247	0,2812	0,0241

Tabla 9.6 Comparación de boosting con otros seis clasificadores.

(1) 50 repeticiones con mfinal=100 y maxdepth=3

(2) 50 repeticiones con mfinal=1000 y maxdeph=3

(3) 100 repeticiones con mfinal=500 y maxdepth=3

Estos resultados confirman, como ya se apuntaba en la comparación de Maclin y Opitz (1997), la superioridad de los clasificadores combinados sobre los individuales, y que el método boosting obtiene mejores resultados que bagging. Pero también ratifica lo complicado que resulta determinar qué clasificador combinado es mejor de manera general, ya que boosting y el bosque aleatorio presentan comportamientos muy similares no pudiendo considerarse como significativas las diferencias existentes entre ambos. Por tanto, se puede concluir que la combinación de clasificadores individuales supera la precisión de los sistemas individuales. Además, el método boosting presenta resultados mejores que el método bagging y tan buenos como el bosque aleatorio.

Además, en términos de variabilidad boosting ocupa la segunda posición en la primera prueba, la tercera plaza en la segunda prueba y la cuarta posición en la última prueba. Estos resultados, por tanto, no confirman la supuesta ganancia en estabilidad del método boosting. También es cierto, por otra parte, que siempre que se habla de inestabilidad de clasificadores como los árboles de clasificación o redes neuronales es ante *pequeños cambios* en el conjunto de entrenamiento que afectarían mucho menos al clasificador boosting. Las distintas muestras que se utilizan en esta comparación se obtienen mediante muestreo aleatorio del conjunto total, manteniendo las proporciones de las clases. Se utiliza el 90% para entrenamiento y el resto para test. Teniendo en cuenta que el conjunto total es de 2.730 en el caso de dos clases y de 1.536 para las tres clases, las diferencias de un conjunto de entrenamiento a otro pueden no ser pequeñas. Por tanto, estos resultados no pueden juzgar la estabilidad o inestabilidad ante pequeños cambios.

9.4.6. Predicción del fracaso empresarial ampliando el horizonte temporal

Hasta ahora se ha tenido en cuenta únicamente la información del último año antes del fracaso, es decir, se ha intentado predecir el fracaso un año antes de que suceda. Sin embargo, parece lógico pensar que el fracaso de una empresa, en la mayoría de las ocasiones, no será algo espontáneo, de un año para otro, sino que será el resultado de un proceso continuado de deterioro. Por tanto, desde la perspectiva interna de la

empresa, cuanto antes se detecten los síntomas de este deterioro, antes se podrán tomar las medidas oportunas para evitarlo. Por otro lado, desde fuera de la empresa, las relaciones que se establecen son, en muchos casos, de duración superior al año, por ejemplo mediante la concesión de un crédito, por tanto será interesante conocer no sólo si la empresa fracasará o no al año siguiente, sino varios años más tarde.

Por este motivo en los trabajos sobre predicción del fracaso empresarial es habitual utilizar también información de varios años antes del fracaso. Lo que no está tan claro es cuántos ejercicios se deben considerar en este proceso de retroceso en el tiempo. Lo más habitual es trabajar con los cinco años anteriores, aunque también hay investigaciones en las que se recogen tres, y otras, aunque estas son la minoría, siete u ocho años. Por tanto, no existe unanimidad en este punto. En este trabajo se ha optado por la elección más habitual de considerar información de hasta cinco años antes.

Otra decisión a adoptar al llegar a este punto es la de trabajar con modelos temporales o atemporales. Por modelo temporal se entiende la delimitación de un clasificador para cada año anterior al fracaso, mientras que modelo atemporal implica entrenar un clasificador con la información del año anterior al fracaso y dicho clasificador es el que se aplica a los valores de los ratios de los diversos ejercicios anteriores. En Rodríguez-Vilariño (1994) se defienden los modelos atemporales ya que, según esta autora, "*al realizar un estudio de posible insolvencia realista no se puede conocer a qué distancia temporal se encuentra de la misma*". Es decir, no se sabría cuál de los modelos temporales habría que aplicar. En la misma línea se manifiesta Somoza (2001), según el cuál, "*los resultados de las funciones para cada ejercicio previo son tan dispares que realmente hacen desconfiar de su posible aplicación*".

Por todo esto, en este trabajo se ha optado por aplicar el clasificador entrenado en el último año previo a los años anteriores. Además, se ha construido un nuevo clasificador global que utilice la información de los cinco años previos al fracaso. Como se ha hecho hasta ahora, se distingue en primer lugar el caso dicotómico y,

posteriormente, se analiza el caso de tres clases. Los resultados para el caso dicotómico se recogen en la tabla 9.7.

		CLASE REAL			
CLASE ESTIMADA	Año 1	Entrenamiento	8,958%	Test	6,569%
		Fracaso	Activa	Fracaso	Activa
	Fracaso	1204	196	135	16
	Activa	24	1032	2	121
	Año 2	Entrenamiento	11,197%	Test	10,584%
		Fracaso	Activa	Fracaso	Activa
	Fracaso	1186	233	129	21
	Activa	42	995	8	116
	Año 3	Entrenamiento	14,495%	Test	13,504%
		Fracaso	Activa	Fracaso	Activa
	Fracaso	1182	310	131	31
	Activa	46	918	6	106
	Año 4	Entrenamiento	30,822%	Test	27,007%
		Fracaso	Activa	Fracaso	Activa
	Fracaso	1180	709	134	71
	Activa	48	519	3	66
	Año 5	Entrenamiento	22,761%	Test	20,438%
		Fracaso	Activa	Fracaso	Activa
	Fracaso	1186	517	135	54
	Activa	42	711	2	83

Tabla 9.7 Matrices de confusión y errores para los distintos años previos en el caso dicotómico

Los datos que se muestran son el resultado de aplicar el modelo entrenado con la información del último año previo al fracaso en los ejercicios anteriores. Se aplica por separado a las empresas que pertenecían al conjunto de entrenamiento y al conjunto de test aunque, una vez más se insiste en que ahora no se vuelve a entrenar el clasificador, sino que se utiliza el mismo que se ha entrenado previamente. En el caso dicotómico, estos resultados confirman la hipótesis inicial de que a medida que aumenta la distancia al momento del fracaso es más difícil predecirlo. El error en el conjunto de test aumenta hasta el 27% en el cuarto año anterior al fracaso, comparado con el 6,57% que conseguía ese mismo modelo aplicado sobre la información para el último año anterior al fracaso. El error en el conjunto de test del 20,44% en el quinto año previo, si bien es inferior al 27% del cuarto año, sigue siendo muy superior a los de los tres ejercicios previos.

C. ESTIMADA	CLASE REAL				
	Modelo global	Entrenamiento Fracaso	0,000% Activa	Test Fracaso	7,664% Activa
	Fracaso	1228	0	131	15
	Activa	0	1228	6	122

Tabla 9.8 Matrices de confusión y errores del modelo global en el conjunto de entrenamiento y de test

El modelo global, por su parte, se comporta bastante bien consiguiendo un error en el conjunto de test del 7,66%. Además, únicamente clasifica a seis empresas fracasadas como activas. Se podría esperar que al incluir más información el error fuese menor que con el modelo que emplea únicamente la información del último año previo (6,57%). Aunque también es cierto que se ha multiplicado por cinco el número de variables explicativas (en concreto pasa de 18 a 82, ya que, a excepción de la forma jurídica y la variable CNAE1, las otras 16 variables se consideran para cada uno de los cinco años anteriores). El aumento tan elevado de variables explicativas puede llevar asociado problemas como la presencia de variables que aportan poca información, la aparición de ruido, o el problema de la maldición de la dimensión. Por todo ello se puede considerar como satisfactorio el resultado del modelo global.

Los resultados para el caso de tres clases se recogen a continuación en la tabla 9.9.

		CLASE REAL					
CLASE ESTIMADA	Año 1	Entrenamiento		24,657%	Test		21,569%
		Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
	Activa	365	3	0	42	0	0
	Fracaso1	67	310	94	5	37	10
	Fracaso2	29	148	367	4	14	41
	Año 2	Entrenamiento		35,719%	Test		33,333%
		Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
	Activa	272	5	0	29	0	0
	Fracaso1	164	289	133	19	38	16
	Fracaso2	25	167	328	3	13	35
	Año 3	Entrenamiento		47,939%	Test		42,484%
		Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
	Activa	115	5	0	11	1	0
	Fracaso1	328	283	139	36	38	12
	Fracaso2	18	173	322	4	12	39
	Año 4	Entrenamiento		39,696%	Test		29,412%
		Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
	Activa	254	8	0	33	1	0
	Fracaso1	189	288	169	16	37	13
	Fracaso2	18	165	292	2	13	38
	Año 5	Entrenamiento		36,298%	Test		32,026%
		Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
	Activa	293	4	1	34	0	0
	Fracaso1	151	304	176	14	37	18
	Fracaso2	17	153	284	3	14	33

Tabla 9.9 Matrices de confusión y errores para los distintos años previos en el caso de tres clases

Al igual que en el caso dicotómico, los datos que se muestran son el resultado de aplicar el modelo entrenado con la información del último año previo a los ejercicios anteriores. Se consideran por separado las empresas que pertenecían al conjunto de entrenamiento y las del conjunto de test. Al considerar las tres clases, el problema aumenta considerablemente en complejidad, como muestra el hecho de que, en el mejor de los casos, los errores son superiores al 20% en el conjunto de test. También es más difícil confirmar la hipótesis inicial de que a medida que aumenta la distancia al momento del fracaso es más difícil predecirlo. El error en el conjunto de test aumenta hasta el 42,48% en el tercer año anterior al fracaso, comparado con el 21,57% que conseguía ese mismo modelo aplicado sobre el último año anterior al fracaso. Sin embargo, en los años cuatro y cinco antes del fracaso, el error en el conjunto de test está

en niveles 29,4% y 32% respectivamente. Por tanto, en los ejercicios entre dos y cinco años antes del fracaso el modelo entrenado para el último año se comporta peor que en este ejercicio. Sin embargo, parece que no existe, en este caso, un deterioro progresivo de la precisión a medida que se aleja el horizonte temporal de predicción.

C. ESTIMADA	CLASE REAL					
Modelo global	Entrenamiento		0,000%	Test		22,876%
	Activa	Fracaso1	Fracaso2	Activa	Fracaso1	Fracaso2
Activa	461	0	0	42	0	0
Fracaso1	0	461	0	6	39	14
Fracaso2	0	0	461	3	12	37

Tabla 9.10 Matrices de confusión y errores en el conjunto de entrenamiento y de test para el modelo global

De nuevo el modelo global, se comporta bastante bien consiguiendo un error en el conjunto de test del 22,87% y un error nulo en el conjunto de entrenamiento. Además, no clasifica a ninguna empresa fracasada como activa. Como ya se ha comentado anteriormente, el pequeño aumento del error del modelo global comparado con el del modelo que incluía únicamente la información del último año previo al fracaso (21,57%), puede justificarse por el aumento considerable de variables explicativas que podrían traer asociadas la aparición de los problemas ya mencionados. Por ello se puede considerar como satisfactorio el resultado del modelo global.

9.4.7. Resumen de la aplicación del método boosting a la predicción del fracaso empresarial

En este capítulo se ha recogido una breve visión teórica de cómo debe tratarse la información antes de su uso en un problema de clasificación. En segundo lugar, se ha descrito el proceso seguido en el muestreo y la depuración de los datos. A continuación, se han aplicado árboles de clasificación y el método boosting a la predicción del fracaso empresarial, considerando por separado la posibilidad de trabajar con dos y tres clases. En cuarto lugar, se ha comparado el método boosting con otros seis sistemas de clasificación, siendo el bosque aleatorio el único método que obtenía resultados casi tan buenos como los de boosting. Por último, se ha aplicado el modelo entrenado con la

información del último año previo a los ejercicios anteriores y, también, se ha construido un modelo utilizando la información de los cinco ejercicios anteriores al año del fracaso. A continuación, la tabla 9.10 recoge de manera muy resumida un esquema de la aplicación práctica realizada. Finalizada la aplicación práctica se aborda a continuación el capítulo de las conclusiones y las líneas de investigación futura.

Pronóstico del fracaso empresarial	Principales resultados
Árbol de clasificación con dos clases	<La poda mejora la precisión del árbol, del 13,87% al 9,124%. <CF.PT1 y PE.PT1 son los dos ratios seleccionados.
Boosting con dos clases	<Reduce el error del 9,124% al 6,57%. <CF.PT1, PE.PT1, BAI.FP1 y BAI.AT1 son los ratios más importantes.
Árbol de clasificación con tres clases	<La poda mejora la precisión del árbol, del 30,72% al 26,14%. <PE.PT1 y lnAT1 son las dos variables seleccionadas.
Boosting con tres clases	<Reduce el error del 26,14% al 21,56%. < PE.PT1, BAI.AT1 y CF.PT1 son los ratios más importantes.
Comparación de Boosting con otros clasificadores	<Los clasificadores combinados superan en precisión a los individuales. <Boosting y Random Forest son los dos mejores métodos.
Ampliación del horizonte temporal	<Deterioro de la capacidad predictiva al alejarse del momento del fracaso. <El modelo global no mejora la precisión del modelo del último año.

Tabla 9.11 Cuadro resumen de la aplicación práctica realizada

CAPÍTULO 10

CONCLUSIONES Y

LÍNEAS DE INVESTIGACIÓN FUTURA

Para finalizar este trabajo de investigación se van a exponer las principales conclusiones que se pueden extraer de los capítulos precedentes, también se dará una escueta visión de las posibles líneas de investigación que regirán el futuro trabajo del doctorando en relación al campo de estudio de este trabajo.

10.1. CONCLUSIONES

A continuación se recogen las principales conclusiones que se desprenden de este trabajo, pudiendo distinguir aquellas que hacen referencia a la parte metodológica de aquellas que corresponden a la parte empírica.

1.- Utilidad y conveniencia de trabajar con combinaciones de clasificadores

La combinación de clasificadores potencia las ventajas de los clasificadores básicos y solventa sus inconvenientes. Como se ha comentado en la primera parte de este trabajo los puntos fuertes y débiles de los sistemas individuales son los siguientes: en el discriminante lineal y el vecino más próximo, al ser procedimientos geométricos, la naturaleza de los datos es importante, debiendo ser variables cuantitativas. El vecino más próximo es el método que más capacidad de almacenamiento necesita y el que más tarda en clasificar una nueva observación. Las Redes Neuronales Artificiales son un método no paramétrico de clasificación, que permite obtener un grado elevado de precisión. Este sistema es el menos comprensible para el usuario por lo que se le asimila a una caja negra. Tanto en las redes como en los árboles de clasificación, la naturaleza de la información es poco relevante, siendo adecuados para datos cualitativos. Los árboles son el procedimiento más fácil de entender para el usuario.

Suponiendo que las tasas de error de los clasificadores individuales son inferiores al cincuenta por ciento y que éstos son independientes, el error cometido por la combinación disminuirá al aumentar el número de clasificadores básicos incluidos. Por otra parte, la combinación ideal consiste en utilizar clasificadores cuanto más precisos mejor y que estén en desacuerdo el mayor número de veces posible, ya que la combinación de clasificadores idénticos no aporta ningún beneficio.

Siguiendo con ese razonamiento, se plantean tres razones para justificar el uso de clasificadores combinados, una estadística, una de computación y una tercera de representación. La primera se basa en la eliminación del riesgo asociado a la elección de un clasificador frente al resto, especialmente en conjuntos de tamaño reducido donde la precisión estimada puede ser similar para varios de ellos. La razón de computación plantea la posibilidad de que los sistemas de clasificación que realizan una búsqueda local puedan quedar atrapados en un óptimo local, y verse imposibilitados para alcanzar el óptimo global. Finalmente, la razón de representación aduce la imposibilidad de representar la verdadera función con ninguna de las hipótesis disponibles.

Finalmente, desde la perspectiva bayesiana también se aboga por la agregación de modelos cuando el objetivo no es la selección explícita de un modelo, sino extraer el mejor rendimiento posible al conjunto de datos disponible para la predicción que se desea realizar.

2.- La combinación de clasificadores reduce el error de generalización

Se ha estudiado el *error de generalización* de los clasificadores individuales, es decir, el error que se espera que cometa cuando se aplique en observaciones nuevas que no estaban presentes en el conjunto de entrenamiento. Este error se ha desagregado en la suma de tres términos no negativos que son el riesgo de Bayes, el sesgo y la varianza. El primero de ellos recoge el error inherente al conjunto de datos y que ningún método de clasificación puede reducir. El sesgo mide el error persistente del método de clasificación, es decir, el error que se mantendría incluso si se tuviese un conjunto infinito de clasificadores, entrenados de manera independiente. El término correspondiente a la varianza mide el error causado por las fluctuaciones que se producen al generar un clasificador individual. Por tanto, a partir de este enfoque la idea que se desprende es que promediando varios clasificadores se puede reducir el término varianza y, de este modo, disminuir también el error esperado o de generalización.

3.- Los clasificadores combinados son más estables

Se entiende por un clasificador inestable aquel que sufre grandes cambios ante pequeñas modificaciones en el conjunto de entrenamiento. En este sentido los árboles de clasificación y las redes neuronales son considerados como métodos inestables, mientras que el análisis discriminante lineal y el método del vecino más próximo se consideran estables. Sin embargo, cuando el tamaño del conjunto de entrenamiento es demasiado pequeño en comparación con la dimensión de los datos, todos los clasificadores sufren problemas de inestabilidad. Es decir, ante estas circunstancias incluso el discriminante lineal es inestable.

En general, los clasificadores más estables se comportan mejor mientras que los clasificadores más inestables obtienen peores resultados en la predicción. Por tanto, el estudio de la estabilidad es interesante si se pretende mejorar la precisión de un clasificador. Diversos autores, por ejemplo Skurichina, han demostrado que los clasificadores combinados, y más concretamente el método boosting, consiguen una mayor estabilidad.

4.- Superioridad del método boosting sobre el método bagging

Los dos métodos construyen los clasificadores básicos en versiones modificadas del conjunto de entrenamiento y los combinan después en la regla de clasificación final, sin embargo, se diferencian en el modo de obtener los clasificadores básicos. En concreto las diferencias fundamentales son tres, en primer lugar, en la versión inicial de adaboost (Freund y Schapire , 1996) se utilizan todas las observaciones en cada paso del boosting (no se aplica bootstraping). Además, la regla de decisión final de bagging no pondera a los clasificadores básicos de manera distinta según su precisión, mientras que en boosting si lo hace. Por último, pero quizás lo más importante, el clasificador obtenido en cada iteración del método boosting depende de todos los anteriores, mientras que en bagging son independientes.

La aplicación realizada en este trabajo ha confirmado los resultados existentes previamente en la literatura, mostrando la superioridad del método boosting sobre el método bagging, como puede comprobarse en el apartado 9.4.5. Boosting supera en precisión a bagging en las tres pruebas recogidas, además se muestra más estable en dos de estas tres pruebas.

5.- El método boosting puede verse como un modelo aditivo generalizado

El interés que presenta la relación entre el método boosting y los modelos aditivos generalizados se basa fundamentalmente en que permite una mejor comprensión del funcionamiento y de los resultados que obtiene boosting.

A pesar de su interesante sencillez, los modelos lineales tradicionales no son adecuados en muchas situaciones, especialmente cuando se trabaja con datos reales. Por ello, se han desarrollado otros métodos más flexibles, como son los modelos aditivos generalizados, donde se sustituye cada término lineal por una función, que en principio no tiene por qué ser paramétrica, y se mantiene la aditividad permitiendo interpretar el modelo de manera muy similar al inicial.

Los modelos aditivos generalizados son, por decirlo de alguna manera, una combinación lineal de funciones que no tienen por qué ser lineales. Los coeficientes de esta combinación lineal se denominan coeficientes de expansión. Pues bien, boosting puede verse como una manera de ajustar una expansión aditiva de un conjunto de funciones básicas. En el caso de boosting, las funciones básicas son los clasificadores individuales y los coeficientes calculados para ponderar a cada clasificador básico en función de su error, actúan como coeficientes de expansión.

6.- Importancia económica de la predicción del fracaso empresarial

En este trabajo ha quedado sobradamente justificada la relevancia económica de la predicción del fracaso empresarial por la gran cantidad de individuos o entidades que se pueden ver afectadas por este hecho, tanto desde el interior de la empresa, como son trabajadores y accionistas, como desde fuera de ésta, inversores y analistas financieros, entidades financieras, clientes, proveedores y otros acreedores, auditores e instituciones públicas como la Agencia Tributaria o la Seguridad Social.

Además, dependiendo de la importancia de la empresa para el sector y para la zona o zonas donde esté ubicada (municipio, región, país) las repercusiones pueden ampliarse a otros niveles. Hoy en día no cabe ninguna duda de que la globalización económica es una realidad, por tanto, se puede suponer que también las repercusiones pueden llegar a ser globales. Ante este panorama las distintas administraciones públicas (locales, regionales, nacionales e incluso internacionales) deben interesarse por prever el fracaso de determinadas empresas y tomar las medidas oportunas que eviten su desaparición.

7.- Propuesta de ampliación del concepto de fracaso empresarial

En la mayoría de trabajos sobre pronóstico del fallo empresarial únicamente se consideran como fracasadas aquellas empresas que han entrado en un proceso concursal de suspensión de pagos o quiebra. Según Lizárraga (1996) esto se debe a que, de esta forma, se consigue trabajar con un concepto riguroso, ajeno a interpretaciones subjetivas y presente en bases de datos asequibles, lo que da mayor objetividad a esa investigación empírica. Pero como se ha comentado en el capítulo ocho, existen otras situaciones en las que la empresa ha fracasado, a parte de la suspensión de pagos y la quiebra.

En la aplicación práctica realizada se ha comprobado la posibilidad de incluir otros tipos de fracaso, considerando para ello también como empresas fracasadas a aquellas entidades que han desaparecido (disueltas) y aquellas otras que han sido absorbidas por otras organizaciones. Aunque no en todos los casos la absorción es un síntoma de fracaso, se ha considerado como tal pues rompe la continuidad de la empresa. Los resultados obtenidos para el caso de dos clases, cuando únicamente se discrimina entre empresas fracasadas y empresas activas, muestran sobradamente que estas empresas presentan un patrón de comportamiento diferenciado. Se logran errores en el conjunto de test del 9,124%, para el árbol podado, y del 6,569%, para el clasificador adaboost.M1.

Además, se ha superado la limitación del caso dicotómico distinguiendo entre activas, fracaso1 (disueltas y absorbidas) y fracaso2 (suspensión de pagos y quiebras). Cuando se amplia el número de clases a tres los errores en el conjunto de test son del 26,144% para el árbol podado y del 21,569% para el clasificador adaboost.M1. Estos resultados muestran la mayor dificultad del problema al ampliar el número de clases. Pero, si se analizan los errores cometidos, puede verse como el solapamiento se produce entre los dos tipos de fracaso y no entre estos y las empresas activas. Esto quiere decir que el patrón de comportamiento de las empresas disueltas y absorbidas es, efectivamente, más parecido al de las empresas en suspensión de pagos y quiebra que al de las empresas activas.

8.- Relevancia de considerar factores cuantitativos y cualitativos en la predicción del fracaso empresarial

La utilidad de los modelos de predicción queda probada por la multitud de trabajos empíricos realizados con ayuda de los mismos. Estos modelos deben entenderse como un instrumento más en la toma de decisiones empresariales. Los resultados obtenidos en la aplicación práctica realizada en este trabajo verifican la utilidad de la información contable, y en particular de los ratios financieros, para identificar situaciones de fracaso y, por lo tanto, para la toma de decisiones relacionadas con éste. Si no se dispusiese de esa información, sería casi imposible predecir estas situaciones.

Los ratios financieros calculados a través de las cuentas de los estados financieros son los criterios más utilizados en la predicción del fracaso empresarial. Sin embargo, no debe obviarse que, normalmente, los ratios financieros sólo son los síntomas de los problemas financieros y de funcionamiento que una empresa atraviesa, pero no la causa de estos problemas. Por tanto, la diversidad de factores que inciden en la insolvencia empresarial hace conveniente considerar conjuntamente aspectos cuantitativos y cualitativos para la predicción del fracaso. En este trabajo se ha incluido como información cualitativa, la forma jurídica de la empresa y el sector de actividad al que pertenece a través del código de la CNAE a un dígito.

9.- No es necesario emparejar las empresas por tamaño y sector

En la predicción del fracaso empresarial es una práctica habitual llevar a cabo el emparejamiento de las empresas por tamaño y sector, para controlar estas características que podrían incidir en la relación entre ratios financieros y fracaso. De esta manera, cada empresa fracasada se une a una empresa activa (*sana*) del mismo sector de actividad, con datos de los mismos años y con un tamaño similar. En relación al sector de actividad económica, en general, el emparejamiento se realiza en función del código CNAE. En cuanto al tamaño, habitualmente se utiliza el activo como variable indicadora

del mismo, pero también pueden utilizarse otras variables como el número de empleados o el volumen de ventas.

Pues bien, en este trabajo aunque se ha mantenido un tamaño similar para las distintas clases, ya sean dos o tres las clases con las que se trabaja, no se ha considerado necesario llevar a cabo un emparejamiento de las empresas, por sector o tamaño. De hecho estas características se han incluido en el análisis como variables explicativas junto a los ratios financieros. Rodríguez-Vilariño (1994) afirma que *"diversos trabajos garantizan que un mismo modelo es aplicable a empresas heterogéneas pero pertenecientes a una misma economía nacional"*. Es decir, cuando el interés del estudio no se centra en un sector determinado sino en una determinada zona geográfica, ya sea provincial, regional, nacional o internacional, la consideración de los distintos sectores proporciona al estudio un mayor potencial de generalización.

La importancia relativa de estas variables en los modelos construidos varía dependiendo de si se trabaja con dos o tres clases. Para el caso dicotómico la importancia relativa del sector, a través de la variable CNAE1, es del 5,73% y la del tamaño, a través del lnAT, del 2,6%. Mientras que al aumentar el número de clases estas variables ganan en importancia alcanzando niveles del 6,79% y 5,4%, respectivamente para el sector y el tamaño.

10.- Importancia del desarrollo de software que facilite la aplicación

En las últimas décadas se han desarrollado un conjunto de técnicas provenientes de la Inteligencia Artificial. Estas técnicas han mostrado en las aplicaciones realizadas que son susceptibles de ser aplicadas a la predicción del fracaso empresarial. Una de sus principales ventajas es la flexibilidad de estos modelos, que no tienen que cumplir unas hipótesis tan restrictivas como los modelos estadísticos tradicionales.

Pues bien, para que las distintas técnicas de la Inteligencia Artificial o el Aprendizaje Automático ayuden de forma efectiva al mundo empresarial, es necesario un proceso de

normalización en el intercambio de conocimiento y terminología utilizada. Pero, sobre todo, es necesario un esfuerzo en el desarrollo de software, y cuanto más sencillo mejor, como requisito indispensable para la expansión de los modelos diseñados al ámbito de la empresa.

Buena parte del tiempo dedicado a este trabajo ha correspondido a la programación de los algoritmos de bagging y adaboost.M1 en lenguaje R. Para cada método se han desarrollado tres funciones, una que construye el clasificador adaboost.M1 (o bagging) y clasifica a las observaciones del conjunto de entrenamiento. Otra función que utiliza el clasificador entrenado para predecir la clase de un conjunto de observaciones nuevas. Finalmente, una tercera función permite aplicar validación cruzada sobre un conjunto de observaciones. Estas funciones pueden verse en el apéndice A.

Cualquier función en R requiere de un conjunto de argumentos iniciales, en el caso de adaboost.M1, estos argumentos son la fórmula donde se especifica cuál es la variable dependiente y cuáles son las independientes. El nombre del objeto (data frame en la terminología de R) que contiene los datos que se quieren utilizar. Si se desea utilizar la técnica de remuestreo o la de reponderación. El número de iteraciones o lo que es lo mismo el número de árboles a utilizar en la combinación. Y, por último, tres parámetros que son propios de la función *rpart* pero que se trasladan también a adaboost.M1 para limitar el tamaño del árbol.

La salida que devuelve esta función es, en terminología de R, una lista que contiene los siguientes aspectos, la fórmula utilizada, todos los árboles construidos, las ponderaciones correspondientes a los distintos árboles, la suma ponderada de los votos que las observaciones reciben para cada clase, la clase que se asigna a cada observación y la importancia relativa de cada variable en la clasificación.

11.- Boosting mejora la precisión de los clasificadores individuales

En la aplicación práctica realizada en este trabajo se ha comprobado que el método boosting consigue incrementar la precisión del clasificador individual en el conjunto de test. Además, esta mejora se produce tanto en el caso dicotómico como en el caso de tres clases.

En primer lugar, se han comparado únicamente el árbol de clasificación construido según el método CART (Breiman, 1984) y el clasificador adaboost.M1. En el caso dicotómico el árbol, debidamente podado, consigue una precisión del 9,124% en el conjunto de test. Adaboost.M1 disminuye el error en el conjunto de test hasta el 6,569%, lo que supone una reducción del 28% comparado con el error del árbol podado. Cuando se dividen las empresas fracasadas en los dos grupos fracaso1 y fracaso2, adaboost.M1 consigue un error en el conjunto de prueba del 21,569%, lo que comparado con el 26,144% del árbol podado individual supone una reducción del 17,5%. Además, analizando las matrices de confusión puede verse como la mayoría de los errores se cometen al clasificar como fracasadas empresas que continúan siendo activas.

También se han incorporado a la comparación otros métodos de clasificación individuales y combinados. Los sistemas individuales son el método del vecino más próximo, el discriminante lineal y las redes neuronales. Los clasificadores combinados utilizados son bagging y el bosque aleatorio. La comparación entre los siete clasificadores se ha realizado tanto para el caso dicotómico como para las tres clases. Los clasificadores combinados se muestran superiores a los individuales y, entre ellos, son boosting y el bosque aleatorio los que mejor se comportan.

12.- Relevancia de los ratios de solvencia y rentabilidad

Los resultados obtenidos en la aplicación práctica realizada en este trabajo de investigación son coherentes con los de los trabajos recopilados en el apartado 8.2.3 del mismo. Así, los ratios que se muestran como más relevantes son ratios de solvencia

(CF.PT1 y PE.PT1) y rentabilidad (BAI.FP1 y BAI.AT1). Este hecho avala la medida de importancia relativa de las variables utilizada en la función adaboost.M1 que se ha programado. Esta medida ayuda a una mejor comprensión del modelo construido.

Cuando se diferencia entre empresas fracasadas y activas, los ratios más destacados son CF.PT1, PE.PT1, BAI.FP1 y BAI.AT1 con una importancia relativa del 15'1, 12'5, 11'2, y 10'2% respectivamente. Cabe destacar que estos cuatro ratios acaparan, prácticamente, el 50% del total de cortes realizados en los cien árboles desarrollados.

Resultados muy similares se obtienen al diferenciar entre las tres clases, activas, fracaso1 y fracaso2. Tres de los cuatro ratios resaltados en el caso anterior continúan siendo los más destacados. Si bien, en este caso el orden de relevancia es distinto En concreto, ahora se muestra más importante el ratio PE.PT1 (23,61%), seguido de los ratios BAI.AT1 (14,92%) y CF.PT1 (11,07%). También debe observarse que sólo entre estos tres ratios aglutinan casi el 50% de todos de cortes realizados en los nodos del conjunto de árboles construidos a lo largo de las mil iteraciones de boosting.

13.- Ampliación del horizonte temporal para el pronóstico del fracaso empresarial

En este trabajo se ha puesto de manifiesto la conveniencia de examinar la situación de una empresa con información no sólo del período inmediatamente anterior al que se pretende evaluar, sino con varios períodos de desfase. Para comprobar el poder explicativo de los ratios con distintos desfases respecto al período que se pretende examinar, se podrían elaborar diferentes modelos para cada uno de los años previos a la situación de fracaso.

Por otro lado, si el objetivo es establecer un modelo capaz de predecir el fracaso a priori, teniendo en cuenta el desconocimiento del año concreto en que éste se producirá, se puede elaborar un modelo que tenga en cuenta los valores de los ratios de varios años consecutivos antes de la crisis considerados conjuntamente. La ventaja de un modelo

de este tipo es que trabaja con mayor cantidad de información, pero el inconveniente es que el horizonte de predicción sigue siendo a un año vista.

Por todo esto, en este trabajo se ha optado por aplicar el clasificador entrenado en el último año previo a los años anteriores. Tanto en el caso dicotómico como para las tres clases, los resultados confirman que al aumentar la distancia al momento del fracaso es más difícil predecirlo. Si bien este deterioro no es lineal y, en algunos casos, en años más alejados se consiguen errores menores que en otros más cercanos. Por ejemplo, en el caso dicotómico, el error en el conjunto de test del 20,44% en el quinto año previo, es inferior al 27% del cuarto año, aunque es muy superior a los de los tres ejercicios previos.

Además, se ha construido un nuevo clasificador global que utilice la información de los cinco años previos al fracaso. Estos modelos construidos con los ratios de los cinco años anteriores al fracaso se comportan, para dos y tres clases, ligeramente peor que los modelos construidos únicamente con información del último año antes del fracaso. Esto puede deberse al aumento elevado de variables explicativas. Este aumento puede llevar asociado problemas como la presencia de variables que aportan poca información, la aparición de ruido, o el problema de la maldición de la dimensión.

Teniendo en cuenta que el horizonte del pronóstico es el mismo, un año antes del fracaso, que los resultados obtenidos son mejores para el modelo del último año previo y que necesita menos información, en este caso, parece preferible utilizar el modelo que trabaja sólo con la información del año anterior al fracaso, en lugar del modelo global.

Una vez expuestas las principales conclusiones de este trabajo y antes de pasar a exponer las líneas de investigación futura que quedan abiertas a raíz de este trabajo, se ha creído conveniente realizar un ejercicio de autocrítica, para reconocer aquellos aspectos de los objetivos iniciales del trabajo que no se han cumplido.

En primer lugar, después de varias décadas de trabajos en predicción del fracaso empresarial, parece evidente que el objetivo inicial de establecer un sistema totalmente automático no se ha alcanzado todavía. Las tasas de error conseguidas en este trabajo del 6,57% y 21,56% para dos y tres clases, respectivamente, parecen aconsejar que esta clasificación se utilice sólo como una prueba más de que es posible que la empresa fracase, en lugar de una prueba concluyente en sí misma. Por tanto, debe entenderse como una señal de alarma que conduzca a un análisis más detallado de la situación de la empresa.

Por otra parte, estos métodos procesan con rapidez grandes cantidades de información y la transforman en evaluaciones bastante fiables acerca de la salud financiera de las empresas. Esto hace de ellos una herramienta muy interesante para ser utilizada como método de preselección de las empresas candidatas a ser estudiadas con un mayor detenimiento. De este modo contribuyen a un mejor aprovechamiento de los recursos.

Un inconveniente que presenta el método boosting, y en general todos los métodos de combinación de clasificadores, es que incrementan la complejidad del clasificador individual. Habitualmente se utilizan los árboles de clasificación como clasificadores básicos del método boosting. Obviamente no es lo mismo interpretar un único árbol que cien árboles. Sin embargo, los árboles utilizados en boosting suelen ser árboles poco desarrollados, y por tanto, más simples. Además, la función adaboost.M1 que se ha programado evita que se pueda considerar boosting como una caja negra. Esta función permite tener acceso a los árboles individuales de las distintas iteraciones y, también, las ponderaciones de estos árboles, es decir, la influencia de cada árbol en el clasificador final.

Por último, y a pesar de que algunos autores como Skurichina (2000) afirman que boosting es un método muy estable, los resultados obtenidos no destacan este hecho. Debe recordarse que se considera que un clasificador es inestable cuando sufre grandes cambios ante pequeñas modificaciones del conjunto de entrenamiento.

En las comparaciones realizadas entre boosting y otros seis métodos de clasificación, se han realizado 50 repeticiones para el caso dicotómico y para una de las pruebas con tres clases. Además, con las tres clases se ha realizado también una prueba con 100 repeticiones. En términos de variabilidad boosting ocupa la segunda posición en la primera prueba, la tercera posición en la segunda prueba y la cuarta posición en la última prueba. Estos resultados, por tanto, no confirman la supuesta supremacía en estabilidad del método boosting. También es cierto, por otra parte, que siempre que se habla de inestabilidad de clasificadores, como los árboles de clasificación o redes neuronales, es ante *pequeños cambios* en el conjunto de entrenamiento que afectarían mucho menos al clasificador boosting. Las distintas muestras que se utilizan en esta comparación se obtienen mediante muestreo aleatorio del conjunto total, manteniendo las proporciones de las clases. Se utiliza el 90% para entrenamiento y el resto para test por lo que, dado el tamaño de los conjuntos, las diferencias de un conjunto de entrenamiento a otro pueden no ser pequeñas. Por tanto, estos resultados no deben juzgar la estabilidad o inestabilidad ante pequeños cambios.

10.2. LÍNEAS DE INVESTIGACIÓN FUTURA

Lejos de suponer un estudio concluso, este trabajo es el inicio y germen de un campo de investigación con un amplio abanico de cuestiones teóricas y prácticas que deben ser analizadas. A continuación se mencionan algunas de la que se consideran más importantes:

i Modificación de algunos aspectos del método boosting

Sería muy interesante realizar algunas modificaciones relativas a diversos aspectos relacionados con el método boosting utilizado en este trabajo. En primer lugar, la implementación en lenguaje R de otros algoritmos de la familia boosting. En especial el algoritmo *arc-x4* propuesto por Breiman (1996b) que llama la atención por su relativa sencillez.

Otra posible modificación de boosting consistiría en trabajar con las probabilidades de pertenencia a cada clase que asigna cada clasificador básico. Normalmente, boosting recoge la clase asignada por cada clasificador sin tener en cuenta el grado de seguridad en esa predicción. Por tanto, sería interesante estudiar los resultados de boosting considerando esa información.

Además, se intentará programar la función *adaboost.M1* para que proporcione una medida de la importancia en base a la ganancia de información que ha conseguido cada variable en las distintas particiones en las que ha sido utilizada. De esta manera se conocerá de manera más realista la importancia de las distintas variables.

Por último, del mismo modo que los árboles individuales se pueden convertir en un conjunto de reglas de decisión, sería interesante sintetizar el conjunto de árboles generados a lo largo de las distintas iteraciones de boosting en un conjunto de reglas.

Incorporación de más información de entrada al problema del pronóstico del fracaso empresarial

Aunque los resultados obtenidos son satisfactorios, no se debe caer en una actitud conformista y, por tanto, hay que intentar mejorarlos. Para intentar mejorar los resultados se puede incluir información adicional, cualitativa y cuantitativa, que pudiera ser importante como la antigüedad de la empresa, el número de empleados, la gestión

de las empresas, su organización, el nicho de mercado al que pertenece la empresa y la posición que ocupa en él, las ventajas competitivas que posee, etc. (Zopounidis, 1998).

Por último se podría considerar una mayor desagregación del concepto de fracaso, considerando otras posibles situaciones de fracaso de una empresa no contempladas en este trabajo. Una vez superada la limitación del caso dicotómico, es posible considerar otras facetas del fracaso que se consideren interesantes.

Utilización de otros clasificadores básicos en boosting

En este trabajo se han utilizado como clasificadores básicos del método boosting los árboles de clasificación. En el futuro se desea estudiar el comportamiento de boosting utilizando otros clasificadores básicos distintos a los árboles. Por ejemplo, entrenar una red neuronal en cada iteración de boosting y combinarlas después, según el error cometido por cada una de ellas.

También sería conveniente estudiar con mayor detenimiento el método del bosque aleatorio, que ha obtenido resultados muy interesantes en la comparación con el método boosting. Analizando cómo influye la elección de los parámetros que controlan su crecimiento, como son el número de variables entre las que elegir la partición de cada nodo o el número de iteraciones a realizar.

Aplicación del método boosting a otros problemas económicos

En primer lugar, se va a desarrollar un análisis similar al planteado en este trabajo pero restringiendo el ámbito geográfico a Castilla-La Mancha. Dicho análisis sería importante dado el carácter marcadamente regional de la Universidad de Castilla- La Mancha, constituyendo una institución estratégica fundamental para el desarrollo de la Comunidad Autónoma de Castilla-La Mancha. Se trataría de establecer un modelo que ayude a pronosticar el fracaso o continuidad de las empresas castellano-manchegas y ver si el patrón de comportamiento es distinto o no al del ámbito nacional.

Los trabajos de pronóstico del fracaso empresarial de ámbito regional han sido incipientes desde la década de los noventa destacando los desarrollados en la Comunidad Valenciana, Gallego, Gómez y Yáñez (1996) y Ferrando y Blanco (1998) y en Galicia, Rodríguez (2002b). Por tanto, se diseñarán modelos específicos adaptados al entorno temporal y geográfico de Castilla-la Mancha para la predicción del fracaso empresarial.

Finalmente, sería interesante la aplicación del método boosting en ámbitos distintos de la economía, donde todavía no se ha aplicado. Existen áreas de la economía donde es habitual el uso de técnicas de aprendizaje supervisado, por ejemplo el credit scoring, y, por tanto, también se puede aplicar boosting.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

ABAD, C.; ARQUERO, J.L. Y JIMÉNEZ, S. (2003): *"Procesos de fracaso empresarial. Identificación y contrastación empírica"*. III Workshop de Investigación Empírica en Contabilidad Financiera. Alicante.

ABNEY, S.; SCHAPIRE, R.E. Y SINGER, Y. (1999): *"Boosting applied to tagging and pp attachment"*. En Proceedings of the Joint SIGDAT Conference on Empirical Methods En Natural Language Processing and Very Large Corpora.

ALFARO, E.; GÁMEZ, M. Y GARCÍA, N. (2002): *"Una revisión de los métodos de clasificación aplicables a la Economía"*. Documento de Trabajo 2/2002/1 de la Facultad de CC. Económicas y Empresariales de Albacete. Universidad de Castilla-La Mancha.

ALFARO, E.; GÁMEZ, M. Y GARCÍA, N. (2003): *"Una revisión de los métodos de agregación de clasificadores"*. Actas de la XVII Reunión Asepelt España, Almería.

ALLWEIN, E.L.; SCHAPIRE, R.E. Y SINGER, Y. (**2000):** "*Reducing multiclass to binary: A unifying approach for margin classifiers"*. Journal of Machine Learning Research, 1:113-141.

ALTMAN, E.I. (1968): *"Financial ratios, discriminant analysis and the prediction of corporate bankruptcy"*. Journal of finance, vol. 23, 4, pp. 589-609.

ALTMAN, E.I. (1993): *"Corporate financial distress and bankruptcy"*. John Wiley and Sons, Nueva York.

ALTMAN, E.I. (2000): *"Predicting financial distress of companies: revisiting the Z-score and zeta models"*. III Jornadas de Predicción de la Insolvencia Empresarial, Torremolinos.

ANDERSON, J.A. (1984): *"Regression and ordered categorical variables"*. J. R. Statist. Soc. B, 46, pp. 1-30.

ARQUES, A. (1997): *"La predicción del fracaso empresarial. Aplicación al riesgo crediticio bancario"*. Tesis Doctoral, Universidad de Murcia .

ARQUES, A.; CALVO-FLORES, A. Y GARCÍA, D. (1997): *"Factores discriminantes del riesgo financiero en la industria manufacturera española"*. En Calvo-Flores, A. y García, D. (Coord.): Predicción de la insolvencia empresarial, pp. 125-156. AECA .

AUDRINO, F. Y BÜHLMANN, P. (2003): *"Volatility estimation with functional gradient descent for very high-dimensional financial time series"*. Journal of Computational Finance, vol. 6, 3, pp. 65-89.

AZOFF, E. (1994): *"Neural networks time series forecasting of financial markets"*. Ed. Wiley & Sons Ltd.

BARNEY, D.K.; GRAVES, O.F. Y JOHNSON, J.D. (1999): *"The farmers home administration and farm debt failure prediction"*. Journal of Accounting and Public Policy, n.18, pp. 99-139.

BARNIV, R.; AGARWAL, A. Y LEACH, R. (1997): *"Predicting the outcome following bankruptcy filing: A three-state classification using neural networks"*. International

Journal of Intelligent Systems En Accounting Finance and Management, vol. 6, 3, pp. 177-194.

BARNIV, R.; AGARWAL, A. Y LEACH, R. (2002): *"Predicting bankruptcy resolution"*. Journal of Business, Finance & Accounting, vol. 29, 3-4, pp. 497-520.

BAUER, E. Y KOHAVI, R. (1999): *"An empirical comparison of voting classification algorithm: Bagging, boosting and variants"*. Machine Learning, vol. 36, pp 105-142.

BEAVER, W.H. (1966): *"Financial ratios as predictors of failure"*. Empirical research in Accounting: Selected Studies. Suplemento al Volumen 5 del Journal of Accounting Research, pp 71-111.

BECKER, R.; CHAMBERS, J. Y WILKS, A. (1988): "*The new S Language"*. Chapman & Hall, Nueva York.

BEGG, C.B. Y GRAY, R. (1984): *"Calculation of polychotomous logistic regression parameters using individualized regressions"*. Biometrika, 71, pp. 11-18.

BERZAL, F.; CUBERO, J.C.; MARÍN, N. Y SÁNCHEZ, D. (2004): "*Building multi-way decision trees with numerical attributes"*. Information Sciences, n. 165, pp. 73-90.

BERZAL, F.; CUBERO, J.C.; SÁNCHEZ, D. Y SERRANO, J.M. (2004): *"ART: A hybrid classification model"*. Machine Learning, vol. 54, n. 1, pp. 67-92.

BISHOP, C. M. (1995): "*Neural networks for pattern recognition*". New York: Oxford University Press Inc.

BLUM, L.A. Y LANGLEY, P. (1997): *"Selection of relevant features and examples in machine learning"*. Artificial Intelligence Journal, special issues on relevance, 97(1-2), pp. 245-271.

BLUM, M. (1974): *"Failing company discriminant analysis"*. Journal of Accounting Research, pp. 1-23.

BONSÓN, E.; ESCOBAR, T. Y MARTÍN, M.P. (1999): *"Aplicación de los sistemas de inducción de árboles de decisión a la gestión empresarial: toma de decisiones y control de tareas de decisión"*. En Bonsón, E. (Ed): Tecnologías Inteligentes para la gestión empresarial, RA-MA editorial, Madrid, pp. 115-132.

BREIMAN, L. (1996a): *"Bagging predictors"*. Machine Learning, vol. 24, 2, pp.123-140.

BREIMAN, L. (1996b): *"Bias, variance, and arcing classifiers"*. Technical Report 460, Statistics Department, University of California.

BREIMAN, L. (1997a): *"Arcing the edge"*. Technical Report 486, Statistics Department, University of California.

BREIMAN, L. (1997b): *"Prediction games and arcing algorithms"*. Technical Report 504, Statistics Department, University of California.

BREIMAN, L. (1998): "Arcing classifiers". The Annals of Statistics, vol. 26, 3, pp. 801-849.

BREIMAN, L. (1999): *"Random forests-random features"*. Technical Report 567, Statistics Department, University of California.

BREIMAN, L. (2000): *"Some infinite theory for predictor ensembles"*. Technical Report 577, Statistics Department, UC Berkeley.

BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R. Y STONE C.J. (1984): *"Classification and regression trees"*. Belmont, Wadsworth International Group.

CALVO-FLORES, A. Y GARCÍA, D. (COORD.) (1997): *"Predicción de la insolvencia empresarial"*. Editado por AECA.

CALVO-FLORES, A. Y GARCÍA, D. (2002): *"Relación entre la posición económica y financiera de la empresa y los estados de fracaso empresarial"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 47-71. AECA.

CASADO, S.; GÓMEZ, O.; NÚÑEZ, L. Y PACHECO, J. (2004): *"El problema de selección de variables. Aplicación a la predicción del riesgo de quiebra de las empresas"*. Actas de la XVIII Reunión Asepelt España, León.

CASADO, S.; NÚÑEZ, L.; GÓMEZ, O. Y PACHECO, J. (2005): *"Predicción de la insolvencia empresarial: uso de búsqueda tabú para selección de ratios explicativos"*. (En proceso de evaluación) Revista de Economía Aplicada.

CASEY, C. Y BARTCZAK, N.(1985): *"Using operating cash flow data to predict financial distress: some extensions"*. Journal of Accounting Research, vol. 23, 1, Spring, pp. 384-401.

CESA-BIANCHI, N.; CONCONI, A. Y GENTILE, C. (2002): *"A second-order perceptron algorithm"*. En Proceedings of the Annual Conference on Computational Learning Theory, vol. 2375 of LNAI, pp. 121-137, Sydney,. Springer.

CHAMBERS, J. Y HASTIE, T. (ED) (1992): *"Statistical models in S"*. Chapman & Hall, Nueva York.

CLARK, L.A. Y PREGIBON, D. (1992): *"Tree-based models"*. Capítulo 9 en Chambers y Hastie (1992).

CLARKE, W.R.; LACHENBRUCH, P.A. Y BROFFITT, B. (1979): *"How nor normality affects the quadratic discriminant function"*. Comm. Statist.- Theory Meth, IT-16, pp. 41-46.

COLLINS, M.; SCHAPIRE, R.E. Y SINGER, Y. (2000): *"Logistic regression, AdaBoost and bregman distances"*. En Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.

CORREA, A.; ACOSTA, M. Y OTROS (2003): *"Factores explicativos del desenlace de la quiebra: aportación empírica"*. XII Congreso AECA. Cádiz

COX. D.R. (1966): *"Some procedures associated with the logistic qualitative response curve"*. En David, F.N. editor, Research papers on statistics: Festschrift for J. Neyman, pp. 55-77. John Wiley, New York.

CRAWLEY, M. J. (2005): *"Statistics: an introduction using R"*. John Wiley & Sons, Chichester.

CRESPO, M. A. (2000): *"Una aproximación a la predicción del fracaso empresarial mediante redes neuronales"*. IX Encuentro de Profesores Universitarios de Contabilidad. Las Palmas de Gran Canaria, pp. 591-607.

DALGAARD, P. (2002): *"Introductory statistics with R"*. Springer, Nueva York.

DAMBOLENA, I. Y KHOURY, S. (1980): *"Ratio stability and corporate failure"*. The Journal of Finance, vol. 35, n. 4, pp.1017-1026.

DAY. N.E. Y KERRIDGE, D.F. (1967): *"A general maximum likelihood discriminant"*. Biometrics, 23, pp. 313-324.

DE LA TORRE, J.M.; MERELO, J.J. Y ROMÁN, I. (2001): *"Forecasting business failure. A comparison of neural networks and logistic regression for the Spanish companies"*. European Accounting Congress, Athens.

DEAKIN, E.B. (1972): *"A discriminant analysis of predictors of business failure"*. Journal of Accounting Research, pp. 167-179.

DETTLING, M. Y BÜHLMANN, P. (2003): *"Boosting for tumor classification with gene expression data"*. Bioinformatics, vol. 19, n. 9, pp. 1061-1069.

DEVIJVER, P.A. Y KITTLER, J.V. (1982): *"Pattern recognition. A statistical approach"*. Prentice Hall- Englewood Cliffs.

DÍAZ, Z. Y FERNÁNDEZ, J. (2003): *"Predicción de crisis empresariales en seguros no vida. Una aplicación del algoritmo See5"*. Documento de Trabajo de la Facultad de CC. Económicas y Empresariales. Universidad de Complutense de Madrid. 2004-10.

DÍAZ, Z.; FERNÁNDEZ, J. Y SEGOVIA, M.J. (2004): *"Sistemas de inducción de reglas y árboles de decisión aplicados a la predicción de insolvencias en empresas aseguradoras"*. Documento de Trabajo de la Facultad de CC. Económicas y Empresariales. Universidad Complutense de Madrid. 2004-09.

DIETTERICH, T.G. (1998): *"An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization"*. Machine Learning, 40 (2), 1–22.

DIETTERICH, T.G. (1999): *"Machine learning research: Four current directions"*. AI Magazine, 18(4):97-136.

DIETTERICH, T.G. (2000): *"Ensemble methods in machine learning"*. En Multiple Classifier Systems, Cagliari, Italia.

DIETTERICH, T.G. Y BAKIRI, G. (1995): *"Solving multiclass learning problems via error-correcting output codes"*. Journal of Artificial Intelligence Research, 2:263-286.

DIMITRAS, A.I.; ZANAKIS, S.H. Y ZOPOUNIDIS, C. (1996): "*A survey of business failures with an emphasis on prediction methods and industrial applications"*. European Journal of Operational Research, 90, pp. 487-513.

DIZDAREVIC, S.; LARRAÑAGA, P. Y OTROS (1997): *"Statistical and machine learning methods in the prediction of bankruptcy"*. International Meeting on Artificial Intelligence in Accounting Finances and Taxes, Huelva, España, pp. 85-100.

DIZDAREVIC, S.; LARRAÑAGA, P.; Y OTROS (1999): *"Predicción del fracaso empresarial mediante la combinación de clasificadores provenientes de la estadística y el aprendizaje automático"*. En Bonsón, E. (Ed): Tecnologías Inteligentes para la gestión empresarial, RA-MA editorial, Madrid

DOMINGOS, P. (1997): *"Why does bagging work? A bayesian account and its implications"*. En David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press.

DRUCKER, H. (1999): *"Boosting using neural nets"*. En A.J.C. Sharkey, editor, Combining Artificial Neural Nets: Ensemble and Modular Learning, pp. 51-77. Springer.

DRUCKER, H. (2000): *"Effect of pruning and early stopping on performance of a boosted ensemble"*. En Proceedings of the International Meeting on Nonlinear Methods and Data Mining, pp. 26-40, Rome, Italy.

DRUCKER, H. Y CORTES, C. (1996): *"Boosting decision trees"*. En D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, Advances in Neural Information Processing Systems 8: Proc. of NIPS'95, vol. 8, pp. 479-485. The MIT Press.

DRUCKER, H.; CORTES, C.; JACKEL, L.D.; LECUN, Y. Y VAPNIK, V. (1994): *"Boosting and other ensemble methods"*. Neural Computation, 6(6):1289-1301.

DRUCKER, H.; SCHAPIRE, R. Y SIMARD, P. (1993): *"Boosting performance in neural networks"*. International Journal of Pattern Recognition and Artificial Intelligence, 7:705 – 719.

DUDA, R.O.; HART, P.E. Y STORK, D.G. (2001): *"Pattern and classification"*. Segunda edición. John Wiley & Sons.

EDMISTER, R.O. (1972): *"An empirical test of financial ratio analysis for small business failure prediction"*. Journal of Financial and Quantitative Analysis, vol. 7, pp. 1477-1493.

EFRON, B. Y TIBSHIRANI, R.(1995): *"Cross-validation and the bootstrap: estimating the error rate of a prediction rule"*. Technical Report, Stanford University.

ELAM, R. (1975). *''The effect of lease data on the predictive ability of financial ratios"*. Accounting Review, 50 (1), pp. 25–43.

ESCUDERO, G.; MÀRQUEZ, L. Y RIGAU, G. (2000): *"Boosting applied to word sense disambiguation"*. En LNAI 1810: Proceedings of the 12th European Conference on Machine Learning, ECML, pp. 129-141, Barcelona, Spain.

EUROPEAN COMISSION (2002): *"Solvency II: Review of work"*. Working paper, MARKT/2536/02 (en http://europa.eu.int/comm/internal_market/insurance/solvency _en.htm.

EUROPEAN COMISSION (2003): *"Design of a future prudential supervisory system in the EU – Recommendations by the Commission Services"*. Working paper, MARKT/2509/03 (en http://europa.eu.int/comm/internal_market/insurance/solvency _en.htm.

FERRANDO, M. Y BLANCO, F. (1998): *"La previsión del fracaso empresarial en la comunidad valenciana: una aplicación de los modelos discriminante y logit"*. Revista Española de Financiación y Contabilidad, 95, pp. 499-540.

FISHER, R.A. (1936): *"The use of multiple measurements in taxonomic problems"*. Annals of Eugenics, 7, pp. 179-188.

FISHER, R.A. (1938): *"The statistical utilisation of multiple measurements"*. Annals of Eugenics, 8, pp. 376-386.

FLAGG, J.C.; GIROUX, G.A,; Y WIGGINS, C.E.; (1991): *"Predicting corporate bankruptcy using failing firms"*. Review of Financial Economics, vol. 1, 1, pp. 67-78.

FLOYD, S. Y WARMUTH, M. (1995): *"Sample compression, learnability, and the Vapnik-Chervonenkis dimension"*. Machine Learning, 21(3):269-304.

FREUND, Y. (1995): *"Boosting a weak learning algorithm by majority"*. Information and Computation, 121(2):256-285.

FREUND, Y. (2001): *"An adaptive version of the boost by majority algorithm"*. Machine Learning, 43(3):293-318.

FREUND, Y. Y OPPER M. (2002): *"Drifting games and Brownian motion"*. Journal of Computer and System Sciences, 64:113-132.

FREUND, Y. Y SCHAPIRE, R.E. (1996a): *"Game theory, on-line prediction and boosting"*. En Proc. 9th Annu. Conf. on Comput. Learning Theory, pp. 325-332. ACM Press, New York, NY.

FREUND, Y. Y SCHAPIRE, R.E. (1996b): *"Experiments with a new boosting algorithm"*. En Proc. 13th International Conference on Machine Learning, pp. 148-146. Morgan Kaufmann.

FREUND, Y. Y SCHAPIRE, R.E. (1997): *"A decision-theoretic generalization of on-line learning and an application to boosting"*. Journal of Computer and System Sciences, 55(1):119-139.

FREUND, Y. Y SCHAPIRE, R.E. (1998): *"Discussion of the paper arcing classifiers"* by Leo Breiman. The Annals of Statistics, 26(3):824-832.

FREUND, Y. Y SCHAPIRE, R.E. (1999): *"A short introduction to boosting"*. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780.

FREUND, Y. Y SCHAPIRE, R.E. (2000): *"Discussion of the paper additive logistic regression: a statistical view of boosting"* by J. Friedman, T. Hastie and R. Tibshirani. The Annals of Statistics, 38(2):391-293.

FREUND, Y.; IYER, R.; SCHAPIRE, R.E. Y SINGER, Y. (1998): *"An efficient boosting algorithm for combining preferences"*. En Proc. 15th International Conference on Machine Learning.

FREUND, Y.; MANSOUR, Y. Y SCHAPIRE, R.E. (2001): *"Why averaging classifiers can protect against overfitting"*. En Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics.

FREUND, Y.; SCHAPIRE, R.E.; SINGER, Y. Y WARMUTH, M.K. (1997): *"Using and combining predictors that specialize"*. En Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 334-343, El Paso, Texas, 4-6.

FRIEDMAN, J. (1989): *"Regularized discriminant analysis"*. Journal of American Statistical Association, 84, pp. 165-175.

FRIEDMAN, J. (1991): *"Multivariate adaptive regression splines"*. Annals of Statistics 19(1), pp. 1-141.

FRIEDMAN, J.H. (1999): *"Greedy function approximation: A gradient boosting machine"*. Technical report, Department of Statistics, Stanford University.

FRIEDMAN, J.; HASTIE, T. Y TIBSHIRANI, R. (1998): *"Additive logistic regression: a statistical view of boosting"*. Technical report, Department of Statistics, Sequoia Hall, Stanford University.

FRYDMAN, H.; ALTMAN, E. Y KAO, D. (1985): *"Introducing recursive partitioning for financial classification: the case of financial distress"*. Journal of Finance, pp. 269-291.

FUKUNAGA, K.(1990): *"Introduction to statistical pattern recognition"*. Segunda edición. San Diego, Academic Press.

GABÁS, F. (1997): *"Predicción de la insolvencia empresarial"*. En Calvo-Flores, A. y García, D. (Coord.): Predicción de la insolvencia empresarial, pp. 11-31. AECA .

GALLEGO, A.M. Y GÓMEZ, A. (2002): *"Efectos marginales de la metodología Multilogic: interpretación y contrastación empírica en los modelos de insolvencia"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 253-288. AECA .

GALLEGO, A.M.; GÓMEZ, J.C. Y YÁÑEZ, L. (1997): *"Modelos de predicción de quiebras en empresas no financieras"*. Actualidad Financiera, mayo, pp. 3-14.

GÁMEZ, M.; ALFARO, E. Y GARCÍA, N. (2003): *"Una clasificación socioeconómica de las regiones europeas mediante mapas de Kohonen"*. Documento de Trabajo 2/2003/1 de la Facultad de CC. Económicas y Empresariales de Albacete. Universidad de Castilla-La Mancha.

GÁMEZ, M. Y GARCÍA, N. (2002): *"Rating de pequeñas y medianas empresas mediante árboles de clasificación"*. Documento de Trabajo 2/2000/2 de la Facultad de CC. Económicas y Empresariales de Albacete. Universidad de Castilla-La Mancha.

GARCÍA, N. (2004): *"Diseño de redes neuronales artificiales para el mercado inmobiliario. Aplicación a la ciudad de Albacete"*. Tesis doctoral dirigida por Dr. D. Jose María Montero Lorenzo, Universidad de Castilla-La Mancha.

GARCÍA, V. Y GARCÍA, D. (2000): *"Decisiones Financieras y Fracaso Empresarial"*. AECA, Madrid.

GATES, G.V. (1972): *"The reduced nearest neighbour rule"*. IEEE Transactions on Information Theory, IT-18-3, pp. 431-433.

GENTRY, J.; NEWBOLD, P. Y WHITFORD, D. (1985): *"Classifying bankrupt firms with funds flow components"*. Journal of Accounting Research , vol. 23, 1, Spring, pp. 146-160.

GILBERT, E.S. (1969): *"The effect of unequal variance covariance matrices on Fisher's linear discriminant function"*. Biometrics, 25, pp. 505-515.

GÓMEZ, O. ; CASADO, S.; NÚÑEZ, L. Y PACHECO, J. (2004): *"The Problem of Variable Selection for Financial Distress: Applying GRASP Metaheuristics"*. Instituto de Empresa, Working Paper WP04-30.

GONZALEZ, J. (2000): *"Las situaciones de dificultad financiera en el marco de la normativa concursal española"*. En García, V. y García, D.: Decisiones Financieras y Fracaso Empresarial, AECA, Madrid.

GRANDVALET, Y. (2004): *"Bagging equalizes influences"*. Machine Learning, vol. 55, pp 251-270.

HAIR, J. F.; ANDERSON, R. E.; TATHAM, R. L. Y BLACK, W. C. (1999): *"Análisis Multivariante"* Quinta edición, Madrid: Prentice Hall Iberia.

HAMER, M. (1983): *"Failure prediction: Sensitivity of classification accuracy to alternative statistical methods and variable sets"*. Journal of Accounting and Public Policy, vol. 2, pp. 289-307.

HANSEN, L. Y SALOMON, P. (1990): *"Neural networks ensembles"*. IEEE Trans. Pattern Analysis and Machine Intelligence, 12, pp. 993-1001.

HART, P. E. (1968): *"The condensed nearest neighbour rule"*. IEEE Transactions on Information Theory, 14, pp. 515-516.

HASSOUN, M. (1995): *"Fundamentals of Artificial Neural Networks"*. Massachusetts: MIT Press.

HASTIE, T.; TIBSHIRANI, R. Y FRIEDMAN, J. (2001): *"The Elements of Statistical Learning: data mining, inference and prediction"*. Springer.

HAUSSLER, D. (1992): *"Decision theoretic generalizations of the PAC model for neural net and other learning applications"*. Information and Computation, vol. 100, 1.

HAYKIN, S. (1994): *"Neural networks. A comprehensive foundation"*. New Jersey: Prentice Hall.

HEBB, D.O. (1949): *"The organization of behavior"*. Ed. Wiley.

HILERA, J.R. Y MARTÍNEZ. V.J. (1995): *"Redes neuronales artificiales. Fundamentos, modelos y aplicaciones"*. Madrid: RA-MA.

HO, T.K. (1998): *"The random subspace method for constructing decision forests"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), pp. 832-844.

IYER, R.D.; LEWIS, D.D.; SCHAPIRE, R.E.; SINGER, Y. Y SINGHAL, A. (2000): *"Boosting for document routing"*. En Proceedings of the Ninth International Conference on Information and Knowledge Management.

JAIN, A.K.; DUIN, R.P.W.Y MAO J._C. (2000): *"Statistical pattern recognition: a review"*. IEEE Trans. Pattern Analysis and Machine Intelligence, 22(1), pp. 4-37.

JIANG, W. (2000): *"Some results on weakly accurate base learners for boosting regression and classification"*. En Proceedings of the First International Workshop on Multiple Classifier Systems, Cagliari, Italy, June 2000., vol. 1857 of Lecture Notes En Computer Science, pp. 87-96. Springer.

JIANG, W. (2001): *"Some theoretical aspects of boosting in the presence of noisy data"*. Technical Report 01-01, Department of Statistics, Northwestern University.

JOHN, G.H.; KOHAVI, R. Y PFLEGER, K. (1994): *"Irrelevant features and the subset selection problem"*. Proceedings of the Eleventh International Conference on Machine Learning, pp. 121-129. New Brunswick, NJ: Morgan Kaufmann.

JOHNSON, R.A. Y WICHERN, D.W. (1998): *"Applied multivariate statistical analysis"*. Cuarta edición, New Jersey: Prentice Hall.

KAY, J. W. Y TITTERINGTON, D. M. (eds.) (1999): *"Statistics and neural networks. Advances at the interface"*. New York: Oxford University Press.

KEASEY, K Y WATSON , R.(1987): *"Non financial symptoms and the prediction of small company failure: a test of Argenti's hypotheses"*. Journal of Business Finance and Accounting, vol. 14 (3), otoño, pp. 335–354.

KENDALL, M.G.; STUART, A. Y ORD, J.K. (1983): *"The advanced theory of statistics"* vol 3, Design and analysis and time series. Capítulo 44. Griffin, Londres, 4ª edición.

KIERS, H.A.L. Y RASSON, J.-H. (eds.) (2000): *"Data analysis, classification, and related methods "*. Berlín: Springer.

KITTLER, J.; HATEF, M.; DUIN, R.P.W. Y MATAS, J. (**1998):** *"On combining classifiers"*. IEEE Trans. Pattern Analysis and Machine Intelligence., 20, pp. 226-238.

KIVINEN J. Y WARMUTH, M. K. (**1999):** *"Boosting as entropy projection"*. En Proc. Computational Learning Theory, COLT'99.

KOH, H.C. Y TAN, S.S. (1999): *"A neural network approach to the prediction of going concern status"*. Accounting and Business Research, vol. 29, n. 3, pp.211-216.

KOHAVI, R. (1995): *"A study of cross-validation and bootstrap for accuracy estimation and model selection"*. In C. S. Mellish, (ed), Proceedings of IJCAI-95, pp. 1137-1143. Morgan Kaufmann.

KOHAVI, R.; PROVOST, F. Y FAWCETT, T. (1998): *"The case against accuracy estimation for comparing induction algorithms"*. In C. S. Mellish, (ed), Proceedings of IJCAI-95, pp. 1137-1143. Morgan Kaufmann.

KOHONEN, T. (1982): "*Self-organized formation of topologically correct feature maps*". *Biological Cybernetics* 43, pp. 59-69.

KOHONEN, T. (1989): *"Self-Organization Maps"*. 2ª ed. Berlín, Heidelberg, New York: Springer-Verlag.

KOLTCHINSKII, V. Y PANCHENKO, D. (2000): "*Bounding the generalization error of neural networks and combined classifiers"*. En Proc. of the Second ICSC Symposium on Neural Computation, Berlin.

KOLTCHINSKII, V.; PANCHENKO, D. Y LOZANO, F. (2001a): "*Further explanation of the effectiveness of voting methods: The game between margins and weights"*. En Proc. COLT.

KOLTCHINSKII, V.; PANCHENKO, D. Y LOZANO, F. (2001b): "*Some new bounds on the generalization error of combined classifiers"*. Advances in Neural Information Processing Systems 13: Proc. of NIPS'2000.

KROGH, A. Y VEDELSBY, J. (1995): "*Neural networks ensembles, cross validation and active learning"*. En Toretzky, D.; Tesauro, G. y Leen, T.(ed): Advances En Neural Information Processing Systems, vol. 7, pp. 107-115. MIT Press, Cambridge, MA.

KUBAT, M. Y MATWIN, S. (1997): *"Addressing the curse of imbalanced training sets: one-sided selection"* Proceedings of the 14th. International Conference on Machine Learning. pp. 179-186. Morgan Kaufmann.

KUNCHEVA, L.I.. (2004): *"Combining pattern classifiers. Methods and algorithms"*. Wiley.

KUNCHEVA, L.I.; BEZDEK, J.C. Y DUIN, R.P.W. **(2001):** *"Decision templates for multiple classier fusion: and experimental comparison"*. Pattern Recognition, 34, pp.299-314.

KUTIN, S. Y NIYOGI, P. (2001): *"The interaction of stability and weakness in adaboost"*. Technical Report TR-2001-30, University of Chicago Department of Computer Science.

KUTIN, S. Y NIYOGI, P. (**2002):** *"Almost-everywhere algorithmic stability and generalization error"*. Technical Report TR-2002-03, Department of Computer Science, The University of Chicago.

LAFFARGA, J. (1999): *"Los modelos de predicción de la insolvencia empresarial: limitaciones y utilidades"*. Boletín AECA, n. 48, pp. 31-38.

LAFFARGA, J. Y MORA, A. **(2002):** *"La predicción del fracaso empresarial. El estado de la cuestión en España"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 25-45. AECA .

LAITINEN, E.K. (1991): "*Financial ratios and different failure processes"*. Journal of Business, Finance and Accounting, vol. 18, 5, pp. 649-673.

LAITINEN, E.K. (1992): "*Prediction of failure of a newly founded firm"*. Journal of Business Venturing, 7, pp. 323-340.

LAITINEN, E.K. (1993): "*Financial predictors for different phases of the failure process"*. Omega, vol. 21, 2, pp. 215-228.

LAM, L. (2000): *"Classifier combinations: implementations and theoretical issues"*. En Kittler, J. y Roli, F. (ed): Multiple Classifier Systems, vol. 1857 de Lecture Notes En Computer Science, pp. 78-86, Cagliari, Italia. Springer.

LAMOTHE, P. Y ARAGÓN, R. (2003): *"Valoración de empresas asociadas a la nueva economía"*. Editorial Pirámide.

LANGLEY, P. (1994): *"Selection of relevant features in machine learning"*. Proceedings of the AAAI Fall Symposium on Relevance. New Orleans, LA: AAAI Press.

LECUN, Y.; JACKEL, L.D. Y OTROS(1995): *"Learning algorithms for classification: A comparison on handwritten digit recognition"*. Neural Networks, pp. 261-276.

LIAW, A. Y WIENER, M. (2005): *"randomForest: Breiman and Cutler's random forests for classification and regression"*. R package version 4.5-2. http://stat-www.berkeley.edu/users/breiman/RandomForests

LIBBY, R. (1975): *"Accounting ratios and the prediction of failure: some behavioural evidence"*. Journal of Accounting Research, pp. 150-161.

LIPPMAN, R. (1987): *"An introduction to computing with neural nets"*. IEEE ASSP Magazine, Vol. 3, nº 4, pp 4-22.

LITTLESTONE, N. Y WARMUTH, M. (1986): *"Relating data compression and learnability"*. Technical report, University of California at Santa Cruz, USA.

LIZARRAGA, F. (1996): *"Modelos multivariantes de predicción del fracaso empresarial: una aplicación a la realidad de la información contable española"*. Tesis Doctoral, Universidad Pública de Navarra.

LIZARRAGA, F. (1998): *"Modelos de previsión del fracaso empresarial: ¿funciona entre nuestras empresas el modelo de Altman de 1968?"*. Revista de Contabilidad, vol. 1, 1, enero-junio, pp.137-164.

LIZARRAGA, F. (2002): *"La utilidad de los modelos de predicción del fracaso en la empresa española a lo largo de la última década"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 219-252. AECA .

LÓPEZ GARCÍA, S. (2000): *"Métodos estadísticos para la estimación de la precisión de un clasificador"*. http://www.aic.uniovi.es/aprendizaje/publicaciones.html

LÓPEZ, D.; MORENO, J. Y RODRÍGUEZ, P. (1994): *"Modelos de previsión del fracaso empresarial: una aplicación a entidades de seguros en España"*. Esic-Market, 84, pp. 83-125.

LÓPEZ, J.; GANDÍA, J.L. Y MOLINA, R. (1998): *"La suspensión de pagos en las pymes: una aproximación empírica"*. Revista Española de Financiación y Contabilidad, vol. 27, 94, pp. 71-97.

LOWE, D.G. (1995): *"Similarity metric learning for a variable-kernel classifier"*. Neural Computation, 7-1, pp 72-85.

LUGOSI, G. Y VAYATIS, N. (2004): *"On the Bayes-risk consistency of regularized boosting methods"*. Annals of Statistics, vol. 32, pp. 30-55.

MACLIN, R. (1998): *"Boosting classifiers regionally"*. En Proc. of AAAI.

MACLIN, R. Y OPITZ, D. (1997): *"An empirical evaluation of bagging and boosting"*. En Proceedings of the Fourteenth National Conference on Artificial Intelligence, pp. 546-551 Cambridge, MA. AAAI Press/MIT Press.

MAINDONALD, J. Y BRAUN, J. (2003): *"Data analysis and graphics using R : an example-based approach"*. Cambridge University Press, Cambridge.

MARGINEANTU, D.D. Y DIETTERICH, T.G. (1997): *"Pruning adaptive boosting"*. En Proc. 14th International Conference on Machine Learning, pp. 211-218. Morgan Kaufmann.

MARIACA, R. (2002): *"Predicción de problemas de crisis y continuidad en empresas bancarias"*. Documento de Trabajo del Instituto de Investigaciones Socioeconómicas de Bolivia, 11/02.

MARKS, S. Y DUNN, O.J. (1974): *"Discriminant functions when covariance matrices are unequal"*. Journal of American Statistical Association, 69, pp. 555-559.

MARTÍN DEL BRÍO, B. Y SANZ MOLINA, A. (1997): *"Redes neuronales y sistemas borrosos"*. Madrid: RA-MA.

MARTÍN PLIEGO, J.; Y RUÍZ-MAYA PÉREZ, L. (2003): *"Fundamentos de probabilidad"*. Editorial Alfa Centauro.

MARTÍN, J.L. (1997): *"Modelos de pronóstico de la insolvencia empresarial"*. En Calvo-Flores, A. y García, D. (Coord.): Predicción de la insolvencia empresarial, pp. 33-49. AECA.

MARTÍN, M.L.; LEGUEY, S. Y SÁNCHEZ, J.M. (1999): *"Solvencia y estabilidad financiera en la empresa de seguros: metodología y evaluación empírica mediante análisis multivariante"*. Cuadernos de la Fundación Mapfre Estudios, 49.

MARTÍNEZ DE LEJARZA, I. (1996): *"Forecasting company failure: neural approach versus discriminant análisis. An application to spanish insurance companies of 80's"*. En Sierra, G. y Bonsón, E. (Eds.): Intelligent systems En accounting and finance, Huelva, pp. 169-186.

MASON, L.; BARTLETT, P. Y BAXTER, J. (1999): *"Direct optimization of margins improves generalization in combined classifiers"*. En Advances in Neural Information Processing Systems 11: Proc. NIPS'1998, pp. 288-294.

MCCULLOCH, W.S. Y PITTS, W. (1943): *"A logical calculus of the ideas immanent in nervous activity"*. Bulletin of Mathematical Biophysics 5, pp.115-133.

MEIR, R. Y RÄTSCH, G. (2003): "*An introduction to boosting and leveraging"*. En S. Mendelson and A. Smola, editors, Advanced Lectures on Machine Learning, LNCS, pp. 119-184. Springer. En press. Copyright by Springer Verlag.

MEIR, R.; MANNOR, S. Y ZHANG, T. (2002): "*The consistency of greedy algorithms for classification*". En Proceedings of the Annual Conference on Computational Learning Theory, vol. 2375 of LNAI, pp. 319-333, Sydney. Springer. Copyright by Springer, Berlin.

MENSAH, Y. (1983): *"The differential bankruptcy predictive ability of specific price level adjustments: some empirical evidence"*. The Accounting Review, vol. 58, n. 2, Apri), pp. 228-246.

MERLER, S.; FURLANELLO, C.; LARCHER, B. Y SBONER, A. (2001): *"Tuning cost-sensitive boosting and its application to melanoma diagnosis"*. En J. Kittler and F. Roli, editors, Proceedings of the 2nd International Workshop on Multiple Classifier Systems MCS2001, vol. 2096 of LNCS, pp. 32-42. Springer.

MERZ, C.J. Y MURPHY, P.M. (1996): *"UCI Repository of Machine Learning Databases"*. Department of Information and Computer Science, University of Califormia, Irvine, CA. (http://www.ics.uci.edu/~mlearn/MLRepository.html)**.**

MICHIE, D., SPIEGELHALTER, D.J. Y TAYLOR, C.C. (eds.) (1994): *"Machine learning, neural and statistical classification"*. Ellis Horwood.

MINSKY, M. Y PAPERT, S. (1969): *"Perceptrons"* Cambridge, MA: MIT Press.

MITCHELL, T.M. (1997): *"Machine learning"*. McGraw Hill.

MOODY'S INVESTORS SERVICE, (2000): *"Moody's three point plot: a new approach to mapping equity fund returns"*. Moody's Investors Service, Nueva York.

MORA, A. (1994): *"Los modelos de predicción del fracaso empresarial: una aplicación empírica del logit"*. Revista Española de Financiación y Contabilidad, 78, enero-marzo, pp. 203-233.

MORENO, P.J. Y LOGAN, B. Y RAJ, B. (2001): *"A boosting approach for confidence scoring"*. En Eurospeech.

MORGAN, J.N.; MESSENGER, R.C. (1973): *"THAID: a sequential search program for the analysis o nominal scale dependent variables"*. Survey Research Center, Institute for Social Research, University of Michigan.

MORGAN, J.N.; SONQUIST, J.A. (1963): *"Problems in the analysis of survey data, and a proposal"*. Journal of the American Statistical Association, 58, pp. 415-434.

OLSHON, J. A. (**1980):** *"Financial ratios and the probabilistic prediction of bankruptcy"*. Journal of Accounting Research, vol. 18, 1, p. 5-12..

ONODA, T.; RÄTSCH, G. Y MÜLLER, K.-R. (**2000):** *"Applying support vector machines and boosting to a non-intrusive monitoring system for household electric appliances with inverters"*. En Proceedings of NC'2000.

OPITZ, D. Y MACLIN, R. (**1999):** *"Popular ensemble methods: An empirical study"*. Journal of AI Research, 11:169-198.

PALEPU, K. G. (1986): "*Predicting takeover targets. A methodological an empirical analysis"*. Journal of Accounting and Economics, vol. 8, pp. 3-35.

PAVLOVIC, V. Y GARG, A. (2001): *"Efficient detection of objects and attributes using boosting"*. En IEEE Conf. Computer Vision and Pattern Recognition - Technical Sketches, Kauai, HI.

PEEL, M.J. Y PEEL, D.A. (1988): *"A multilogit approach to predicting corporate failure - some evidence for the U.K. Corporote sector"*. Omega, vol. 16, 4, pp. 309-319.

PERRONE, M.P. Y COOPER, L.N. (1993): "*When networks disagree: Ensemble method for neural networks"*. En R.J. Mammone, editor, Neural Networks for Speech and Image processing. Chapman-Hall.

PINA, V. (1989): *"La información contable en la predicción de la crisis bancaria 1977-1985"*. Revista Española de Financiación y Contabilidad, n. 58, pp.309-338.

PLATT, J.D. Y PLATT, M.B. (1990): *"Development of a class of stable predictive variables: the case of bankruptcy prediction"*. Journal of Business, Finance and Accounting, vol. 17, n. 1, pp.31-51.

QUINLAN, J.R. (1979): *"Discovering rules by induction from large collections of examples"*. En "Expert Systems in the Microelectronic Age", ed. D. Michie. Edinburgh: Edinburgh University Press.

QUINLAN, J.R. (1983): *"Learning efficient classification procedures and their application to chess end-games"*. En "Machine Learning", eds. R.S. Michalski, J.G. Carbonell y T.M. Mitchell, pp. 463-482. Palo Alto: Tioga.

QUINLAN, J.R. (1986): *"Introduction of decision trees"*. Machine Learning, 1, pp. 81-106.

QUINLAN, J.R. (1993): *"C4.5: Programs for machine learning"*. San Mateo CA: Morgan Kaufmann.

QUINLAN, J.R. (1996a): *"Bagging, boosting, and C4.5"*. En Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, pp. 725-730, Menlo Park. AAAI Press / MIT Press.

QUINLAN, J.R. (1996b): *"Boosting first-order learning"*. En S. Arikawa and A.K. Sharma (ed.), Proceedings of the 7th International Workshop on Algorithmic Learning Theory, vol. 1160 of LNAI, pp. 143-155, Berlin. Springer.

QUINLAN, J.R. (1998): *"Miniboosting decision trees"*. Journal of AI Research.

R DEVELOPMENT CORE TEAM (2004): *"R: A language and environment for statistical computing"***.** R Foundation for Statistical Computing, Viena. http://www.R-project.org .

RÄTSCH, G. Y WARMUTH, M.K. (2005): "*Efficient margin maximization with boosting*". Submitted to Journal of Machine Learning Research.

RÄTSCH, G.; ONODA, T. Y MÜLLER, K.-R. (1999): *"Regularizing AdaBoost"*. En M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, Advances in Neural Information Processing Systems 11:Proc. of NIPS'98, pp. 564-570. MIT Press.

RÄTSCH, G.; ONODA, T. Y MÜLLER, K.-R. (2001): *"Soft margins for AdaBoost"*. Machine Learning, 42(3):287-320.

RÄTSCH, G.; SCHÖKOPF, B. Y OTROS (2002): *"Ensemble learning in the presence of outliers"*. En D.A. Cohn M.S. Kearns, S.A. Solla, editor, Advances in Neural Information Processing Systems 12: Proc. of NIPS'99. MIT Press.

REFENES, P. (editor) (1995): *"Neural networks in the capital markets"*. Chichester: John Wiley & Sons Ltd.

RENCHER, A.C. (1995): *"Methods of multivariate analysis"*. Nueva York, Wiley.

RIPLEY, B.D. (1993): *"Statistical aspects of neural networks"* En Barndorff-Nielsen, O.; Cox, D.; Jensen, J.; y Kendall, W. (Editores) Chaos and networks-statistical and probabilistic aspects. Chapman and Hall.

RIPLEY, B.D. (1996): *"Pattern Recognition and Neural Networks"* Cambridge, Cambridge University Press.

RIPLEY, B.D. (2004): "*rpart: Recursive Partitioning*". R package version 3.1-20.

RITTER, G.L.; WOODRUFF, H.B.; LOWRY, S.R. Y ISENHOUR, T.L. (1975): *"An algorithm for a selective nearest neighbour decision rule"* IEEE Transactions on Information Theory, 21-6, pp. 665-669.

RIVERO, P. (2002): *"Análisis de balances y estados complementarios"*. Pirámide, Madrid.

RIVERO, P.; BANEGAS, R. Y OTROS (1998): *"Análisis por ratios de los estados contables financieros: (análisis externo)"*. Civitas, Madrid.

RODRIGUEZ, J.J. Y ALONSO, C.J. (**2001):** *"Learning classification radial basis function networks by boosting"*. En J. Kittler and F. Roli, editors, Proceedings of the 2nd International Workshop on Multiple Classifier Systems MCS2001, vol. 2096 of LNCS, pp. 32-42. Springer.

RODRÍGUEZ, M. (2001a): *"Predicción del fracaso empresarial en compañías no financieras. Consideración de técnicas de análisis multivariante de corte paramétrico"*. Actualidad Financiera, junio, pp. 27-42.

RODRÍGUEZ, M. (2001b): *"El fracaso empresarial en galicia a través del análisis tradicional de la información contable. La capacidad predictiva del análisis univariante"*. II Congreso de Economía de Galicia. Santiago de Compostela.

RODRÍGUEZ, M. (2002a): *"La inteligencia artificial como herramienta de modelización en la predicción de la insolvencia empresarial"*. Revista de la Asociación Española de Contabilidad y Administración de Empresas, sep-dic, pp. 36-40.

RODRÍGUEZ, M. (2002b): *"Modelos de insolvencia en empresas gallegas. Aplicación de técnicas paramétricas y de inteligencia artificial"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 73-114. AECA .

RODRÍGUEZ, M. (2004): *"Fundamentos de la inteligencia artificial y sus diferentes aplicaciones como herramienta de modelización en el pronóstico de la insolvencia empresarial"*. Análisis Financiero Internacional, enero-marzo, 115, pp. 25-49.

RODRÍGUEZ-VILARIÑO, M.L. (1994): *"Utilidad del análisis de ratios para la predicción de la insolvencia empresarial"*. Actualidad Financiera, 34-35 y 36, C699-C773.

ROSENBLATT, F. (1959): *"The perceptron: a probabilistic model for information storage and organization in the brain"*. *Psychological Review* 65, pp. 386-408.

RUÍZ-MAYA PÉREZ, L. Y MARTÍN PLIEGO, J. (2004): *"Fundamentos de inferencia estadística"*. Tercera edición. Thompson.

RUIZ-MAYA PÉREZ, L.; MARTÍN PLIEGO, J.; MONTERO LORENZO, J.M. Y URÍZ TOMÉ, P. (1995): *"Análisis estadístico de encuestas: datos cualitativos"*. Editorial: Alfa Centauro (A.C.)

RUMELHART, D. E., HINTON, G. H. Y WILLIAMS, R. J. (1986a): "Learning internal representations by error propagation". *Parallel Distributed Processing: explorations in the Microstructures of Cognition*, Vol. 1, D.E. Rumelhart and J.L. McClelland (Editors.), Cambridge, MA: MIT Press, pp. 318-362.

RUMELHART, D. E., HINTON, G. H. Y WILLIAMS, R. J. (1986b): "Learning representations by back-propagating errors". *Nature* 323, pp. 533-536.

SALZBERG, S.L. (1999): "*On comparing classifiers: a critique of current research and methods"*. Data mining and knowledge discovery, 1, pp. 1-12.

SANCHÍS, A.; GIL, J.A. Y HERAS, A. (2003): *"El análisis discriminante en la previsión de la insolvencia en las empresas de seguros no vida"*. Revista Española de Financiación y Contabilidad, 116, enero-marzo, pp.183-233.

SCHAPIRE, R.E. (1990): *"The strength of weak learnability"*. Machine Learning, 5(2):197-227.

SCHAPIRE, R.E. (1992): *"The design and analysis of efficient learning algorithms"*. MIT Press.

SCHAPIRE, R.E. (1997): *"Using output codes to boost multiclass learning problems"*. En Machine Learning: Proceedings of the Fourteenth International Conference, pp. 313-321.

SCHAPIRE, R.E. (1999a): *"Theoretical views of boosting"*. En Computational Learning Theory: Fourth European Conference, EuroCOLT'99.

SCHAPIRE, R.E. (1999b): *"A brief introduction to boosting"*. En Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.

SCHAPIRE, R.E. (1999c): *"Theoretical views of boosting and applications"*. En Tenth International Conference on Algorithmic Learning Theory.

SCHAPIRE, R.E. (2001): *"Drifting games"*. Machine Learning, 43(3), pp. 265-291.

SCHAPIRE, R.E. (2002): *"The boosting approach to machine learning: An overview"*. En Workshop on Nonlinear Estimation and Classification. MSRI.

SCHAPIRE, R.E.; FREUND, Y.; BARTLETT, P. Y LEE, W.S. (1998): *"Boosting the margin: A new explanation for the effectiveness of voting methods"*. The Annals of Statistics, 26(5):1651-1686.

SCHAPIRE, R.E. Y SINGER, Y. (1999): *"Improved boosting algorithms using confidence-rated predictions"*. Machine Learning, 37(3):297-336.

SCHAPIRE, R.E. Y SINGER, Y. (2000): *"Boostexter: A boosting-based system for text categorization"*. Machine Learning, 39(2/3):135-168.

SCHAPIRE, R.E.; SINGER, Y. Y SINGHAL, A. (1998): *"Boosting and Rocchio applied to text filtering"*. En Proceedings of the 21st Annual International Conference on Research and Development En Information Retrieval.

SCHWENK, H. Y BENGIO, Y. (2000): *"Boosting neural networks"*. Neural Computation, 12(8):1869-1887.

SCOTT, J. (1981): *"The probability of bankruptcy: a comparison of empirical predictions and theoretical models"*. Journal of Banking and Finance, pp. 317-344.

SEGOVIA, M.J. (2003): *"Predicción de crisis empresariales en seguros no vida mediante la metodología rough set"*. Tesis Doctoral, Universidad Complutense de Madrid.

SEGOVIA, M.J.; GIL, J.A. Y OTROS (2003): *"La metodología rough set frente al análisis discriminante en los problemas de clasificación multiatributo"*. XI Jornadas de ASEPUMA, Oviedo.

SERRANO, C. (1994): *"Las redes neuronales artificiales en el análisis de la información contable"*. Tesis Doctoral, Universidad de Zaragoza.

SHARPE, W. (1998): *"Morningstar's risk adjusted ratings"*. Financial Analysts Journal, Julio/Agosto, pp. 21-33.

SKURICHINA, M. (2000): *"Stabilizing weak classifiers"*. Tesis Doctoral, Delft University of Technolog, Delft, Holanda.

SKURICHINA, M. Y DUIN, R.P.W (**2001):** *"Bagging and the random subspace method for redudant feature spaces"*. En J. Kittler and F. Roli, editors, Proceedings of the 2nd Internationa Workshop on Multiple Classifier Systems MCS2001, vol. 2096 of LNCS, pp. 1-10. Springer.

SOLLICH, P. Y KROGH, A. (**1996)**: *"Learning with ensembles: How overfitting can be useful"*. En D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, Advances En Neural Information Processing Systems 8: Proc. of NIPS'95, pp. 190-196. The MIT Press.

SOMOZA, A. (2001): *"La consideración de factores cualitativos, macroeconómicos y sectoriales en los modelos de predicción de la insolvencia empresarial. Su aplicación al sector textil y confección de Barcelona (1994-1997)"*. Papeles de Economía Española, 89-90, pp. 402-426.

SOMOZA, A. (2002): *"Modelos de predicción de la insolvencia: la incorporación de otro tipo de variables"*. En Doldán, F. y Rodríguez, M. (Coord.): La gestión del riesgo de crédito. Métodos y modelos de predicción de la insolvencia empresarial, pp. 139-173. AECA .

STANDARD & POOR'S RATING SERVICES, (2000): *"Money market fund criteria"*. Standard & Poor's, Nueva York.

TAFFLER, R. (1982): *"Finding those firms in danger"*. Accountancy age, 16.

TAPIA, E.; GONZALEZ, J.C. Y VILLENA, J. (**2001):** *"A generalized class of boosting algorithms based on recursive decoding models"*. En J. Kittler and F. Roli, editors,

Proceedings of the 2nd International Workshop on Multiple Classifier Systems MCS2001, vol. 2096 of LNCS, pp. 22-31. Springer.

THEODOSSIOU, P.; KAHYA, E.; SAIDI, R. Y PHILIPPATOS, G. (1996): *"Financial distress and corporate acquisitions: Further empirical evidence"*. Journal of Business Finance and Accounting, 23/5-6, pp. 699-719.

TIBSHIRANI, R.(1996): *"Bias, variance and prediction error for classification rules"*. Technical Report, Stanford University.

TOMÀS, J.; AMAT, O. Y ESTEVE, M. (1999): *"Cómo analizan las entidades financieras a sus clientes"*. Gestión 2000, Barcelona.

TOMEK, I. (1976): *"An experiment with the edited nearest neighbour rule"*. IEEE Transactions on Systems, Man and Cybernetics, 6-6, pp. 448-452.

TRESP, V. (2000): *"A bayesian committee machine"*. Neural Computation, 12(11):2719-2741.

TRESP, V. (2001): *"Committee machines"*. En Handbook on neural Network Signal Processing. CRC Press.

TRIPPI Y TURBAN. (1996): *"Neural networks in finance and investing"*. Ed. Irwin.

URIEL, E. (1995): *"Análisis de datos: series temporales y análisis multivariante"*. Editorial AC.

VALENTINI, G. Y MASULLI, F. (2002): *"Ensembles of learning machines"*. En Marinaro, M. y Tagliaferri, R. (ed) Neural Nets WIRN Vietri, Series Lecture Notes En Computer Sciences, Springer-Verlag, Heidelberg, Alemania.

VENABLES, W. N. Y RIPLEY, B. D. (2002): "*Modern Applied Statistics with S"*. Springer, Nueva York.

WEBB, G.I. (**2000):** *"Multiboosting: A technique for combining boosting and wagging"*. Machine Learning, 40(2):1.

WEISS, S.M. Y KAPOULEAS, I. (1989): *"An empirical comparison of pattern recognition, neural nets and machine learning classification methods (vol 1)"*. En IJCAI89: proceedings of the eleventh international joint conference on artificial intelligence, pp. 781-787. Detroit, Morgan Kaufmann.

WEISS, S.M. Y KULIKOWSKI, C. (1991): *"Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems"*. Morgan Kaufmann Publishers.

WILCOX, J.W. (1971): *"A simple theory of financial ratios as predictors of failure"*. Journal of Accounting Research, otoño, pp. 389-395.

WILSON, D.L. (1972): *"Asymptotic properties of nearest neighbour rules using edited data"* IEEE Transactions on Systems, Man and Cybernetics, 2-3, pp. 408-421.

WILSON, D.R. Y MARTÍNEZ, T.R. (2000): *"Reduction techniques for instance-based learning algorithms"*. Machine Learning, 38(3), pp. 257-286.

WILSON, R.L. Y SHARDA, R. (1994): *"Bankruptcy prediction using neural network"*. Decision Support Systems, 11, pp. 545-557.

WILSON, R.L. Y SHARDA, R. (2000): *"Bankruptcy prediction using neural networks"*. En Trippi, R.R. y Turban, E. (Ed): Neural Networks En Finance and Investing, McGraw-Hill, pp. 367-394.

WOLPERT, D.H. Y MACREADY, W.G. (1999): *"An efficient method to estimate baggings' generalization error"*. Machine Learning, 35:41.

ZAVGREN, C.V. (1983): *"The prediction of corporate failure: the state of the art"*. Journal of Accounting Literature, vol. 1, pp. 1-38.

ZHOU, Z.-H.; WU, J. Y TANG, W. (2002): *"Ensembling neural networks: many could be better than all"*. Artificial Intelligence, 137(1-2):239-263.

ZMIJEWSKI, M. (1984): *"Methodological issues related to the estimation of financial distress prediction models"*. Journal of Accounting Research, vol. 22, pp. 59-86.

ZOPOUNIDIS, C. (1987): *"A multicriteria decision making methodology for the evaluation of the risk of failure and an application"*. *F*oundations of control engineering, 12/1, 45-67.

ZOPOUNIDIS, C. Y DIMITRAS, A.I. (1998): "*Multicriteria decision aid methods for the prediction of business failure"*. Kluwer academic publishers, Dordrecht.

APÉNDICE

APÉNDICE A:

ALGORITMOS DE CLASIFICACIÓN

A.1. FUNCIÓN ADABOOST.M1

```
  adaboost.M1 <- function(formula, data,boos=TRUE, mfinal=100,
maxdepth=nlevels(vardep), minsplit=20, coeflearn="Breiman", cp=0.01)
{


      #Exigimos que coeflearn sea uno de esos dos valores
      if (!(as.character(coeflearn) %in% c("Freund","Breiman"))){
            stop("coeflearn tiene que ser 'Freund' o 'Breiman' ")
      }
      formula<- as.formula(formula)
      vardep <- data[,as.character(formula[[2]])]
           n <- length(data[,1])
```

```
      nclases <- nlevels(vardep)

      pesos <- rep(1/n,n)
      data<-data.frame(pesos, data) #Los pesos en rpart deben ser
una columna del dataframe

            arboles <- list() #Creamos una lista para guardar los
arboles
      pond <- rep(0,mfinal) # Un vector donde guardaremos la
ponderación de cada árbol.

      for (m in 1:mfinal) {
      #Creamos muestras boostrap utilizando los pesos

            if (boos==TRUE) {
                  boostrap<- sample(1:n,replace=TRUE,prob=pesos)
                  fit <-
rpart(formula,data=data[boostrap,-1],maxdepth=maxdepth,
minsplit=minsplit, cp=cp)
                  flearn <-
predict(fit,newdata=data[,-1],type="class")
                  ind<-as.numeric(vardep != flearn)  #crear un
vector indicador
                  err<- sum(ind)/n                       #calcula el
error en esa iteración
            }

            if (boos==FALSE) {
                  data$pesos <- pesos
                  fit <- rpart(formula=formula, weights=data$pesos,
data=data[,-1],maxdepth=maxdepth)
                  flearn <- predict(fit,data=data[,-1],
type="class")
                  ind<-as.numeric(vardep != flearn)  #Crear un
vector indicador
                  err<- sum(pesos*ind)         #Calcula el error
ponderado en esa iteración


            }
            # Diferenciamos entre Freund y Breiman.
```

```
            c<- log((1-err)/err)
            if (coeflearn=="Breiman"){
                  c<- (1/2)*c
            }

            pesos <- pesos*exp(c*ind)
            pesos<- pesos/sum(pesos)

            #Si el error no es menor que la regla por defecto los
pesos se inicializan
            # Según Opitz y Maclin si no 0<err<0.5 se ajustan los
valores de c.


maxerror<-min(1-max(summary(vardep))/sum(summary(vardep)), 0.5)

            if (err>=maxerror) {
            pesos <- rep(1/n,n)
            c<-0.001
            }
            if (err==0) {
            pesos <- rep(1/n,n)
            c<-3
            }

            arboles[[m]] <- fit                #Guardamos los
arboles
            pond[m]<- c                        #Guardamos las
ponderaciones

            #para conocer la importancia de las variables
            if(m==1){summary(fit$fram[,1])->acum}
            else{summary(fit$fram[,1])->acum1
            acum<-acum+acum1
            }

      }

      pred<- data.frame(rep(0,n))

      for (m in 1:mfinal) {
```

```
            if(m==1){pred <-
predict(arboles[[m]],data[,-1],type="class")}
            else{pred <-
data.frame(pred,predict(arboles[[m]],data[,-1],type="class"))}
      }

      classfinal <- array(0, c(n,nlevels(vardep)))
      for (i in 1:nlevels(vardep)){
            classfinal[,i] <-
matrix(as.numeric(pred==levels(vardep)[i]),nrow=n)%*%as.vector(pond)
      }

      predclass <- rep("O",n)
      for(i in 1:n){
            predclass[i] <-
as.character(levels(vardep)[(order(classfinal[i,],decreasing=T)[1])]
)
      }
      #normalizar la importancia de las variables
      acum<-acum[-1]/sum(acum[-1])*100

      output<- list(formula=formula, arboles=arboles,
ponderaciones=pond,votos=classfinal,clase=predclass,
importancia=acum, maxerror=maxerror)

  }
```

A.2. FUNCIÓN BOOSTING.CV

```
  boosting.cv<-function ( formula, data,v=10,boos=TRUE
,mfinal=100,maxdepth=nlevels(vardep), minsplit=5,
coeflearn="Breiman")
  {
      vardep<-data[,as.character(formula[[2]])]
      n <- length(vardep)
      #para validación cruzada 2<v<n
      if(v>n) stop(" el valor de v no es adecuado")
      if(v<2) stop(" el valor de v no es adecuado")
```

```
      predclass <- rep("O",n)

       for (i in 1:v) {
           test <- v * (0:floor(n/v)) + i
           test <- test[test < n + 1]
           fit <- adaboost(formula, data[-test,],boos
,mfinal,maxdepth, minsplit, coeflearn)
      fit.predict<-predict.boosting(fit, data[test,])
           predclass[test] <- fit.predict[[1]]
       }

     # para que devuelva la matriz de confusión
      tabla <- table(predclass, vardep, dnn=c("Clase estimada",
"Clase real"))

      # Para que devuelva el error en newdata
      error<- 1- sum(predclass== vardep)/n

      output<- list(predicción=predclass, confusión=tabla,
error=error)

  }
```

A.3. FUNCIÓN PREDICT.BOOSTING

```
  predict.boosting<- function(object, newdata) {

      vardep <- newdata[,as.character(object[[1]][[2]])]
      mfinal<-length(object[[2]])
      n <- length(newdata[,1])
      nclases <- nlevels(vardep)
      pesos <- rep(1/n,n)
      newdata<-data.frame(newdata,pesos)
      pond<- object[[3]]
      pred<- data.frame(rep(0,n))

      # Crea un dataframe para guardar las pred, al 1º está vacío,
pero luego se va añadiendo
      for (m in 1:mfinal) {
            if(m==1){pred <-
predict(object[[2]][[m]],newdata,type="class")}
```

```
            else{pred <-
data.frame(pred,predict(object[[2]][[m]],newdata,type="class"))}
      }

      classfinal <- array(0, c(n,nlevels(vardep)))
      for (i in 1:nlevels(vardep)){

classfinal[,i]<-matrix(as.numeric(pred==levels(vardep)[i]),nrow=n)%*
%pond
      }

      predclass <- rep("O",n)
      for(i in 1:n){
            predclass[i] <-
as.character(levels(vardep)[(order(classfinal[i,],decreasing=T)[1])]
)
      }
      # para que devuelva la matriz de confusión
      tabla <- table(  predclass,vardep,dnn=c( "Clase estimada",
"Clase real"))

      # Para que devuelva el error en newdata
      error<- 1- sum(predclass== vardep)/n

      output<- list(predicción=predclass, confusión=tabla,
error=error)
  }
```

A.4. FUNCIÓN BAGGING

```
  bagging <- function(formula, data,
mfinal=100,maxdepth=(nlevels(vardep)), minsplit=5, cp=0.01) {

      formula<- as.formula(formula)
      vardep <- data[,as.character(formula[[2]])]
           n <- length(data[,1])
      nclases <- nlevels(vardep)
```

```
            arboles <- list() #Creamos una lista para guardar los
arboles
      replicas <- array(0, c(n,mfinal))

      for (m in 1:mfinal) {

            boostrap<- sample(1:n,replace=TRUE)
            fit <-
rpart(formula,data=data[boostrap,],maxdepth=maxdepth,
minsplit=minsplit, cp=cp)
              flearn <- predict(fit,newdata=data,type="class")
            ind<-as.numeric(vardep != flearn)  #crear un vector
indicador
            err<- sum(ind)/n                        #calcula el error
en esa iteración

            arboles[[m]] <- fit                #Guardamos los
arboles
            replicas[,m]<-boostrap

      #para conocer la importancia de las variables
            if(m==1){summary(fit$fram[,1])->acum}
            else{summary(fit$fram[,1])->acum1
            acum<-acum+acum1
            }

      }

      pred<- data.frame(rep(0,n))
      # Crea un dataframe para guardar las pred, al 1º está vacío,
pero luego se va añadiendo
      for (m in 1:mfinal) {
            if(m==1){pred <-
predict(arboles[[m]],data,type="class")}
            else{pred <-
data.frame(pred,predict(arboles[[m]],data,type="class"))}
      }

      classfinal <- array(0, c(n,nlevels(vardep)))
      for (i in 1:nlevels(vardep)){
```

```
            classfinal[,i] <-
matrix(as.numeric(pred==levels(vardep)[i]),nrow=n)%*%rep(1,mfinal)
      }

      predclass <- rep("O",n)
      for(i in 1:n){
            predclass[i] <-
  as.character(levels(vardep)[(order(classfinal[i,],decreasing=T)[1]
)])
      }

      #normalizar la importancia de las variables
      acum<-acum[-1]/sum(acum[-1])*100
      output<-
list(formula=formula,arboles=arboles,votos=classfinal,clase=predclas
s, replicas=replicas, importancia=acum)

  }
```

A.5. FUNCIÓN BAGGING.CV

```
  bagging.cv<-function ( formula, data,v=10,
mfinal=100,maxdepth=nlevels(vardep), minsplit=5)
  {
      vardep<-data[,as.character(formula[[2]])]
      n <- length(vardep)
      #para validación cruzada 2<v<n
      if(v>n) stop(" el valor de v no es adecuado")
      if(v<2) stop(" el valor de v no es adecuado")

      predclass <- rep("O",n)

      for (i in 1:v) {
          test <- v * (0:floor(n/v)) + i
          test <- test[test < n + 1]
          fit <- bagging(formula, data[-test,],mfinal,maxdepth,
minsplit)
      fit.predict<-predict.bagging(fit, data[test,])
          predclass[test] <- fit.predict[[1]]
      }
```

```
     # para que devuelva la matriz de confusión
      tabla <- table(predclass, vardep, dnn=c("Clase estimada",
"Clase real"))

      # Para que devuelva el error en newdata
      error<- 1- sum(predclass== vardep)/n

      output<- list(predicción=predclass, confusión=tabla,
error=error)

  }
```

A.6. FUNCIÓN PREDICT.BAGGING

```
  predict.bagging<- function(object, newdata) {

      vardep <- newdata[,as.character(object[[1]][[2]])]
      mfinal<-length(object[[2]])
      n <- length(newdata[,1])
      nclases <- nlevels(vardep)

      pred<- data.frame(rep(0,n))

      # Crea un dataframe para guardar las pred, al 1º está vacío,
pero luego se va añadiendo
      for (m in 1:mfinal) {
            if(m==1){pred <-
predict(object[[2]][[m]],newdata,type="class")}
            else{pred <-
data.frame(pred,predict(object[[2]][[m]],newdata,type="class"))}
      }

      classfinal <- array(0, c(n,nlevels(vardep)))
      for (i in 1:nlevels(vardep)){
            classfinal[,i]<-
matrix(as.numeric(pred==levels(vardep)[i]),nrow=n)%*%rep(1,mfinal)
      }

      predclass <- rep("O",n)
      for(i in 1:n){
```

```
            predclass[i] <-
as.character(levels(vardep)[(order(classfinal[i,],decreasing=T)[1])]
)
      }
      # para que devuelva la matriz de confusión
      tabla <- table(predclass, vardep, dnn=c("Clase estimada",
"Clase real"))

      # Para que devuelva el error en newdata
      error<- 1- sum(predclass== vardep)/n
      output<- list(predicción=predclass, confusión=tabla,
error=error)
  }
```

A.7. FUNCIÓN MARGINS

```
  margins<- function(object, newdata) {

      #newdata debe ser el mismo que con el que se construye
adaboost.M1

      vardep <- newdata[,as.character(object[[1]][[2]])]
      n <- length(newdata[,1])
      nclases <- nlevels(vardep)
      votos<- object[[4]]
      votos/apply(votos,1,sum)->votosporc

      margen<-rep(0,n)

      for (i in 1:n) {

      k<-votosporc[i, as.numeric(vardep[i])]-votosporc[i,]
      margen[i]<- min(k[k!=0])

      }

      output<- list( margen=margen)
  }
```

A.8. AYUDA DE LA FUTURA LIBRERÍA DE R

Como se ha comentado anteriormente, se pretende utilizar todas estas funciones programadas para crear una librería de R, que se pueda poner a disposición de la comunidad científica en internet. Esta librería necesita de un manual de ayuda que sería el siguiente

adaboost.M1 **Clasificación utilizando Boosting.**

Descripción

adaboost.M1 implementa el algoritmo adaboost.M1 propuesto por Freund y Schapire en 1996 utilizando como clasificadores básicos árboles de clasificación.

Uso

adaboost.M1(formula, data, boos=TRUE, mfinal=100, maxdepth=nlevels(vardep), minsplit=5, coeflearn="Breiman", cp=0.01)

Argumentos

formula Una fórmula como en la función lm.

data Un data frame que contenga las variables mencionadas en la fórmula.

boos Si TRUE (por defecto), se realiza una muestra bootstrap del conjunto de entrenamiento utilizando los pesos de cada observación en esa iteración. Si FALSE, se utilizan todas las observaciones con sus correspondientes pesos.

mfinal Es el número de iteraciones, o el número de árboles a utilizar. Por defecto es 100.

maxdepth Fija la profundidad máxima de cualquier nodo del árbol, siendo cero la profundidad del nodo raíz. Por defecto se iguala al número de clases.

minsplit Es el número mínimo de observaciones que deben existir en un nodo, para intentar un nuevo corte. Por defecto se fija en 5.

cp Parámetro complejidad. Cualquier corte que no consiga una ganancia de información superior a cp será rechazado.

coeflearn Si "Breiman" (por defecto) se utiliza el coeficiente de actualización de los pesos propuesto por este autor. Si "Freund" se usa el de Freund y Schapire.

Salida

formula La fórmula utilizada.

arboles Los árboles construidos en las distintas iteraciones.

ponderaciones Es un vector con las ponderaciones de los árboles en las distintas iteraciones.

votos Es una matriz que recoge para cada observación el número de árboles que asignan esa observación a cada una de las clases multiplicado por la ponderación correspondiente al árbol.

clase Es la clase asignada por el clasificador final.

importancia Mide la importancia relativa de cada variable en la clasificación. Calculada a partir del número de veces que se utiliza como criterio para realizar un corte.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Referencias

ALFARO, E.; GÁMEZ, M. Y GARCÍA, N. (2003): *"Una Revisión de los métodos de agregación de clasificadores"*. Actas de la XVII Reunión Asepelt España, Almería.

BREIMAN, L. (1998): "Arcing classifiers". The Annals of Statistics, Vol 26, 3, p. 801-849.

FREUND, Y. Y SCHAPIRE, R.E. (1996): *"Experiments with a New Boosting Algorithm"*. En Proceedings of the Thirteenth International Conference on Machine Learning, p. 148-156, Morgan Kaufmann.

Ver también

predict.boosting, boosting.cv

Ejemplos.

```
library(rpart)

data(iris)

names(iris)<-c("LS","AS","LP","AP","Especies")

lirios.adaboost <- adaboost.M1(Especies~LS +AS +LP+ AP, data=iris, boos=T, mfinal=10)


data(kyphosis)

kypho.adaboost <- adaboost.M1(Kyphosis ~ Age + Number + Start, data=kyphosis, mfinal=15)
```

boosting.cv **Validación cruzada con v-particiones con Boosting.**

Descripción

Los datos se dividen en *v* subconjuntos mutuamente excluyentes de aproximadamente el mismo tamaño y se aplica adaboost.M1 sobre (*v-1*) de los subconjuntos. Después, se aplica el clasificador boosting entrenado en el subconjunto restante para predecir la clase de sus ejemplos. Este proceso se repite para cada uno de los *v* subconjuntos.

Uso

boosting.cv (formula, data, v=10, boos=TRUE, mfinal=100, maxdepth=nlevels(vardep), minsplit=5, coeflearn="Breiman")

Argumentos

formula Una fórmula como en la función bagging.

data Un data frame que contenga las variables mencionadas en la fórmula.

v Un número entero, especificando el tipo de validación cruzada. Por defecto es 10. Si se iguala al número de observaciones se realiza la estimación dejando uno fuera.

boos Si TRUE (por defecto) se realiza una muestra bootstrap del conjunto de entrenamiento utilizando los pesos de cada observación en esa iteración. Si FALSE, se utilizan todas las observaciones con sus correspondientes pesos.

mfinal Es el número de iteraciones, o el número de árboles a utilizar. Por defecto es 100.

maxdepth Fija la profundidad máxima de cualquier nodo del árbol, siendo cero la profundidad del nodo raíz. Por defecto se iguala al número de clases.

minsplit Es el número mínimo de observaciones que deben existir en un nodo, para intentar un nuevo corte. Por defecto se fija en 5.

coeflearn Si “Breiman” (por defecto) se utiliza el coeficiente de actualización de los pesos propuesto por este autor. Si “Freund” se usa el de Freund y Schapire.

Salida

predicción Es la clase asignada por el clasificador final.

confusión La matriz de confusión, que compara la clase real con la estimada.

error Devuelve el error.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Ver también

adaboost.M1, predict.boosting

Ejemplos.

```
library(rpart)

data(iris)

names(iris)<-c("LS","AS","LP","AP","Especies")

lirios.boostcv <- boosting.cv(Especies ~ ., v=10, data=iris, mfinal=10, maxdepth=3)


data(kyphosis)

kyphosis.boostcv <- boosting.cv(Kyphosis ~ Age + Number + Start, data=kyphosis,
mfinal=15)
```

predict.boosting **Predicción con Boosting en un conjunto de prueba.**

Descripción

Clasifica un conjunto de observaciones utilizando un objeto boosting previamente entrenado.

Uso

predict.boosting(object, newdata)

Argumentos

object Un objeto que contiene un modelo ajustado de clase boosting. Se asume que es el resultado de alguna función que produce un objeto cuyos componentes tienen el mismo nombre que los devueltos por la función boosting.

newdata Un data frame que contenga los valores que se quieren predecir. Las variables a las que se hace referencia en el lado derecho de la fórmula (object) deben estar presentes con el mismo nombre en newdata.

Salida

predicción Es la clase asignada por el clasificador final.

confusión La matriz de confusión, que compara la clase real con la estimada.

error Devuelve el error.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Ver también

adaboost.M1, boosting.cv.

Ejemplos.

library(rpart)

```
data(iris)

names(iris)<-c("LS","AS","LP","AP","Especies")

sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))

lirios.adaboost <- adaboost.M1(Especies ~ ., data=iris[sub,], mfinal=10)

lirios.predboosting<- predict.boosting(lirios.boosting, newdata=iris[-sub,])


data(kyphosis)

kyphosis.adaboost <- adaboost.M1(Kyphosis ~ Age + Number + Start, data=kyphosis, mfinal=15)

kyphosis.predboosting <- predict.boosting(kyphosis.adaboost, newdata=kyphosis)
```

bagging **Clasificación utilizando Bagging.**

Descripción

bagging implementa el algoritmo bagging propuesto por Breiman en 1996 utilizando como clasificadores básicos árboles de clasificación.

Uso

```
bagging(formula, data, mfinal=100,maxdepth=nlevels(vardep), minsplit=5)
```

Argumentos

formula Una fórmula como en la función lm.

data Un data frame que contenga las variables mencionadas en la fórmula.

mfinal Es el número de iteraciones, o el número de árboles a utilizar. Por defecto es 100.

maxdepth fija la profundidad máxima de cualquier nodo del árbol, siendo cero la profundidad del nodo raíz. Por defecto se iguala al número de clases.

minsplit Es el número mínimo de observaciones que deben existir en un nodo, para intentar un nuevo corte. Por defecto se fija en 5.

cp Parámetro complejidad. Cualquier corte que no consiga una ganancia de información superior a cp será rechazado.

Salida

formula La fórmula utilizada.

arboles Los árboles construidos en las distintas iteraciones.

votos Es una matriz que recoge para cada observación el número de árboles que asignan esa observación a cada una de las clases.

clase Es la clase asignada por el clasificador final.

replicas Las muestras bootstrap utilizadas en cada iteración.

importancia Mide la importancia relativa de cada variable en la clasificación. Calculada a partir del número de veces que se utiliza como criterio para realizar un corte.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Referencias

ALFARO, E.; GÁMEZ, M. Y GARCÍA, N. (2003): *"Una Revisión de los métodos de agregación de clasificadores"*. Actas de la XVII Reunión Asepelt España, Almería.

BREIMAN, L. (1996): *"Bagging predictors"*. Machine Learning, Vol 24, 2, p.123-140.

BREIMAN, L. (1998): *"Arcing classifiers"*. The Annals of Statistics, Vol 26, 3, p. 801-849.

Ver también

predict.bagging, bagging.cv

Ejemplos.

```
library(rpart)

data(iris)
```

```
names(iris)<-c("LS","AS","LP","AP","Especies")

lirios.bagging <- bagging(Especies~LS +AS +LP+ AP, data=iris, mfinal=10)


data(kyphosis)

kypho.bagging <- bagging(Kyphosis ~ Age + Number + Start, data=kyphosis,
mfinal=15)
```

bagging.cv **Validación cruzada con v-particiones con Bagging.**

Descripción

Los datos se dividen en *v* subconjuntos mutuamente excluyentes de aproximadamente el mismo tamaño y se aplica bagging sobre (*v-1*) de los subconjuntos. Después, se aplica el clasificador bagging entrenado en el subconjunto restante para predecir la clase de sus ejemplos. Este proceso se repite para cada uno de los *v* subconjuntos.

Uso

```
bagging.cv ( formula, data, v=10, mfinal=100, maxdepth=nlevels(vardep),
minsplit=5)
```

Argumentos

formula	Una fórmula como en la función bagging.
data	Un data frame que contenga las variables mencionadas en la fórmula.
v	Un número entero, especificando el tipo de validación cruzada. Por defecto es 10. Si se iguala al número de observaciones se realiza la estimación dejando uno fuera.
mfinal	Es el número de iteraciones, o el número de árboles a utilizar. Por defecto es 100.
maxdepth	Fija la profundidad máxima de cualquier nodo del árbol, siendo cero la profundidad del nodo raíz. Por defecto se iguala al número de clases.
minsplit	Es el número mínimo de observaciones que deben existir en un nodo, para intentar un nuevo corte. Por defecto se fija en 5.

Salida

predicción Es la clase asignada por el clasificador final.

confusión La matriz de confusión, que compara la clase real con la estimada.

error Devuelve el error.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Ver también

bagging, predict.bagging.

Ejemplos.

```
library(rpart)

data(iris)

names(iris)<-c("LS","AS","LP","AP","Especies")

lirios.baggingcv <- bagging.cv(Especies ~ ., v=10, data=iris, mfinal=10,maxdepth=3)


data(kyphosis)

kyphosis.baggingcv <- bagging.cv(Kyphosis ~ Age + Number + Start, data=kyphosis, mfinal=15)
```

predict.bagging **Predicción con Bagging en un conjunto de prueba.**

Descripción

Clasifica un conjunto de observaciones utilizando un objeto bagging previamente entrenado.

Uso

```
predict.bagging(object, newdata)
```

Argumentos

object Un objeto que contiene un modelo ajustado de clase bagging. Se asume que es el resultado de alguna función que produce un objeto cuyos componentes tienen el mismo nombre que los devueltos por la función bagging.

newdata Un data frame que contenga los valores que se quieren predecir. Las variables a las que se hace referencia en el lado derecho de la fórmula (object) deben estar presentes con el mismo nombre en newdata.

Salida

predicción Es la clase asignada por el clasificador final.

confusión La matriz de confusión, que compara la clase real con la estimada.

error Devuelve el error.

Autores

Esteban Alfaro Cortés (Esteban.Alfaro@uclm.es) y Matías Gámez Martínez (Matías.Gamez@uclm.es)

Ver también

bagging, bagging.cv.

Ejemplos.

```
library(rpart)

data(iris)

names(iris)<-c("LS","AS","LP","AP","Especies")

sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))

lirios.bagging <- bagging(Especies ~ ., data=iris[sub,], mfinal=10)

lirios.predbagging<- predict.bagging(lirios.bagging, newdata=iris[-sub,])
```

```
data(kyphosis)

kyphosis.bagging <- bagging(Kyphosis ~ Age + Number + Start, data=kyphosis,
mfinal=15)

kyphosis.predbagging <- predict.bagging(Kyphosis ~ Age + Number + Start,
newdata=kyphosis)
```

APÉNDICE B:
SALIDA DE LA FUNCIÓN ADABOOST.M1

B.1. SALIDA DE LA FUNCIÓN ADABOOST.M1

Para mostrar el tipo de salida que devuelve la función Adaboost.M1 se utiliza como ejemplo la salida de esta función con los datos utilizados en el caso dicotómico. Dado el elevado número de observaciones del conjunto de entrenamiento, 2456, algunos componentes de la salida se mostrarán únicamente para una parte de las observaciones. También para abreviar se recogen únicamente los veinticinco primeros árboles del total de cien árboles utilizados. La salida completa puede verse en el CD que se adjunta a este trabajo.

```
>sabi.boosting <- adaboost.M1(ESTADO ~ ., data=sabi[ind,], mfinal=100,
boos=T,cp=0.01)

< sabi.boosting$arboles

[[1]]

n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1180 Activa (0.480456026 0.519543974)
  2) CF.PT1< 0.3629495 1459  282 Fracaso (0.806716929 0.193283071) *
  3) CF.PT1>=0.3629495 997    3 Activa (0.003009027 0.996990973) *

[[2]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1026 Activa (0.4177524 0.5822476)
   2) CF.PT1< 0.4915537 1670  644 Fracaso (0.6143713 0.3856287)
     4) BAI.AT1< -0.02730503 278   31 Fracaso (0.8884892 0.1115108) *
     5) BAI.AT1>=-0.02730503 1392  613 Fracaso (0.5596264 0.4403736)
      10) JURIDICA=Otros,SA 649  206 Fracaso (0.6825886 0.3174114) *
      11) JURIDICA=SL 743  336 Activa (0.4522207 0.5477793) *
   3) CF.PT1>=0.4915537 786    0 Activa (0.0000000 1.0000000) *

[[3]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1099 Activa (0.44747557 0.55252443)
   2) V.AT1>=0.2526088 1757  678 Fracaso (0.61411497 0.38588503)
```

```
     4) CF.PT1< 0.01386104 443   67 Fracaso (0.84875847 0.15124153) *
     5) CF.PT1>=0.01386104 1314  611 Fracaso (0.53500761 0.46499239)
      10) BAI.FP1>=0.6871849 189   36 Fracaso (0.80952381 0.19047619) *
      11) BAI.FP1< 0.6871849 1125  550 Activa (0.48888889 0.51111111) *
   3) V.AT1< 0.2526088 699   20 Activa (0.02861230 0.97138770)
     6) BAI.AT1< 0.01431591 20    5 Fracaso (0.75000000 0.25000000)
      12) CNAE1=1,3,5,7 13    0 Fracaso (1.00000000 0.00000000) *
      13) CNAE1=4 7    2 Activa (0.28571429 0.71428571) *
     7) BAI.AT1>=0.01431591 679    5 Activa (0.00736377 0.99263623) *

[[4]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1199 Fracaso (0.5118078 0.4881922)
   2) CF.PT1< 0.5109423 1917  660 Fracaso (0.6557121 0.3442879)
     4) T.AT1< 0.06994119 1115  292 Fracaso (0.7381166 0.2618834)
       8) PE.PT1< 3.490952 1093  270 Fracaso (0.7529735 0.2470265) *
       9) PE.PT1>=3.490952 22    0 Activa (0.0000000 1.0000000) *
     5) T.AT1>=0.06994119 802  368 Fracaso (0.5411471 0.4588529)
      10) V.AT1>=2.836892 123   20 Fracaso (0.8373984 0.1626016) *
      11) V.AT1< 2.836892 679  331 Activa (0.4874816 0.5125184) *
   3) CF.PT1>=0.5109423 539    0 Activa (0.0000000 1.0000000) *

[[5]]
n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1217 Fracaso (0.5044788 0.4955212)
   2) PE.PT1>=-0.004377676 2039  800 Fracaso (0.6076508 0.3923492)
     4) CNAE1=1,6,9 336   62 Fracaso (0.8154762 0.1845238) *
     5) CNAE1=0,2,3,4,5,7,8 1703  738 Fracaso (0.5666471 0.4333529)
      10) AC.AT1>=0.691848 1035  366 Fracaso (0.6463768 0.3536232) *
      11) AC.AT1< 0.691848 668  296 Activa (0.4431138 0.5568862) *
   3) PE.PT1< -0.004377676 417    0 Activa (0.0000000 1.0000000) *

[[6]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1171 Activa (0.4767915 0.5232085)
   2) PE.PT1>=0.04539606 2143  972 Fracaso (0.5464302 0.4535698)
     4) CNAE1=0,1,2,3,4,6,7,8,9 1277  483 Fracaso (0.6217698 0.3782302)
       8) PE.PT1< 1.530185 1229  444 Fracaso (0.6387307 0.3612693) *
       9) PE.PT1>=1.530185 48    9 Activa (0.1875000 0.8125000) *
     5) CNAE1=5 866  377 Activa (0.4353349 0.5646651)
      10) FM.AT1< -0.03009587 156   44 Fracaso (0.7179487 0.2820513) *
      11) FM.AT1>=-0.03009587 710  265 Activa (0.3732394 0.6267606) *
   3) PE.PT1< 0.04539606 313    0 Activa (0.0000000 1.0000000) *

[[7]]
n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1107 Activa (0.45073290 0.54926710)
   2) PE.PT1>=-0.004377676 2217 1107 Activa (0.49932341 0.50067659)
     4) V.FP1>=33.28917 76    2 Fracaso (0.97368421 0.02631579) *
     5) V.FP1< 33.28917 2141 1033 Activa (0.48248482 0.51751518)
      10) BAI.FP1< -0.6299878 59    0 Fracaso (1.00000000 0.00000000) *
      11) BAI.FP1>=-0.6299878 2082  974 Activa (0.46781940 0.53218060) *
   3) PE.PT1< -0.004377676 239    0 Activa (0.00000000 1.00000000) *

[[8]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1180 Activa (0.4804560 0.5195440)
  2) PE.PT1>=-0.03748473 2202 1022 Fracaso (0.5358765 0.4641235) *
  3) PE.PT1< -0.03748473 254    0 Activa (0.0000000 1.0000000) *

[[9]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 697 Activa (0.2837948 0.7162052)
   2) BAI.FP1< -0.6299878 37   0 Fracaso (1.0000000 0.0000000) *
```

```
   3) BAI.FP1>=-0.6299878 2419 660 Activa (0.2728400 0.7271600)
     6) BAI.FP1>=1.343445 43   5 Fracaso (0.8837209 0.1162791)
      12) BAI.AT1< 0.4595535 35   0 Fracaso (1.0000000 0.0000000) *
      13) BAI.AT1>=0.4595535 8   3 Activa (0.3750000 0.6250000) *
     7) BAI.FP1< 1.343445 2376 622 Activa (0.2617845 0.7382155)
      14) V.AT1>=3.605644 23   0 Fracaso (1.0000000 0.0000000) *
      15) V.AT1< 3.605644 2353 599 Activa (0.2545686 0.7454314) *

[[10]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 805 Activa (0.3277687 0.6722313)
   2) BAI.AT1< -0.1456322 51   6 Fracaso (0.8823529 0.1176471)
     4) T.AT1< 0.2111353 43   0 Fracaso (1.0000000 0.0000000) *
     5) T.AT1>=0.2111353 8   2 Activa (0.2500000 0.7500000) *
   3) BAI.AT1>=-0.1456322 2405 760 Activa (0.3160083 0.6839917)
     6) CF.PT1< 0.5109423 2270 760 Activa (0.3348018 0.6651982)
      12) PE.PT1< 0.2969898 129  39 Fracaso (0.6976744 0.3023256) *
      13) PE.PT1>=0.2969898 2141 670 Activa (0.3129379 0.6870621) *
     7) CF.PT1>=0.5109423 135   0 Activa (0.0000000 1.0000000) *

[[11]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

```
 1) root 2456 855 Activa (0.3481270 0.6518730)
   2) BAI.FP1< -0.6567918 35   0 Fracaso (1.0000000 0.0000000) *
   3) BAI.FP1>=-0.6567918 2421 820 Activa (0.3387030 0.6612970)
     6) CF.PT1< 0.6040436 2305 820 Activa (0.3557484 0.6442516)
      12) CF.PT1>=0.2084229 59  14 Fracaso (0.7627119 0.2372881) *
      13) CF.PT1< 0.2084229 2246 775 Activa (0.3450579 0.6549421) *
     7) CF.PT1>=0.6040436 116   0 Activa (0.0000000 1.0000000) *

[[12]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 934 Activa (0.3802932 0.6197068)
   2) lnAC1>=7.182234 904 444 Activa (0.4911504 0.5088496)
     4) V.FP1>=9.332194 160  41 Fracaso (0.7437500 0.2562500)
       8) T.AT1>=0.001646409 142  27 Fracaso (0.8098592 0.1901408) *
       9) T.AT1< 0.001646409 18   4 Activa (0.2222222 0.7777778) *
     5) V.FP1< 9.332194 744 325 Activa (0.4368280 0.5631720) *
   3) lnAC1< 7.182234 1552 490 Activa (0.3157216 0.6842784)
     6) BAI.FP1< -0.7204996 25   0 Fracaso (1.0000000 0.0000000) *
     7) BAI.FP1>=-0.7204996 1527 465 Activa (0.3045187 0.6954813)
      14) IN.AT1>=3.807062 18   0 Fracaso (1.0000000 0.0000000) *
      15) IN.AT1< 3.807062 1509 447 Activa (0.2962227 0.7037773) *

[[13]]
n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 993 Activa (0.40431596 0.59568404)
   2) CF.PT1< 0.5109423 2342 993 Activa (0.42399658 0.57600342)
     4) V.FP1>=33.52029 56   2 Fracaso (0.96428571 0.03571429) *
     5) V.FP1< 33.52029 2286 939 Activa (0.41076115 0.58923885)
      10) lnAC1>=7.154165 835 407 Fracaso (0.51257485 0.48742515) *
      11) lnAC1< 7.154165 1451 511 Activa (0.35217092 0.64782908) *
   3) CF.PT1>=0.5109423 114   0 Activa (0.00000000 1.00000000) *

[[14]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1093 Activa (0.44503257 0.55496743)
   2) BAI.AT1< 0.01299309 492  180 Fracaso (0.63414634 0.36585366)
     4) CNAE1=0,2,5,6,9 329   89 Fracaso (0.72948328 0.27051672)
       8) lnAC1>=5.39846 264   53 Fracaso (0.79924242 0.20075758) *
       9) lnAC1< 5.39846 65   29 Activa (0.44615385 0.55384615) *
     5) CNAE1=1,3,4,7 163   72 Activa (0.44171779 0.55828221)
      10) CF.PT1< 0.05886818 115   46 Fracaso (0.60000000 0.40000000) *
      11) CF.PT1>=0.05886818 48    3 Activa (0.06250000 0.93750000) *
   3) BAI.AT1>=0.01299309 1964  781 Activa (0.39765784 0.60234216)
     6) V.AT1>=0.233787 1807  773 Activa (0.42778085 0.57221915)
      12) BAI.AT1>=0.09501956 551  221 Fracaso (0.59891107 0.40108893) *
      13) BAI.AT1< 0.09501956 1256  443 Activa (0.35270701 0.64729299) *
```

```
     7) V.AT1< 0.233787 157    8 Activa (0.05095541 0.94904459) *


[[15]]
n= 2456


node), split, n, loss, yval, (yprob)
      * denotes terminal node


 1) root 2456 1163 Activa (0.47353420 0.52646580)
   2) CF.PT1< 0.02580143 695  258 Fracaso (0.62877698 0.37122302)
     4) BAI.AT1< 0.1714501 673  236 Fracaso (0.64933135 0.35066865)
       8) CNAE1=0,1,2,5,7,8,9 571  172 Fracaso (0.69877408 0.30122592) *
       9) CNAE1=3,4,6 102   38 Activa (0.37254902 0.62745098) *
     5) BAI.AT1>=0.1714501 22    0 Activa (0.00000000 1.00000000) *
   3) CF.PT1>=0.02580143 1761  726 Activa (0.41226576 0.58773424)
     6) CF.PT1< 0.5109423 1686  726 Activa (0.43060498 0.56939502)
      12) V.AT1>=3.278404 44    3 Fracaso (0.93181818 0.06818182) *
      13) V.AT1< 3.278404 1642  685 Activa (0.41717418 0.58282582) *
     7) CF.PT1>=0.5109423 75    0 Activa (0.00000000 1.00000000) *


[[16]]
n= 2456


node), split, n, loss, yval, (yprob)
      * denotes terminal node


 1) root 2456 1203 Fracaso (0.51017915 0.48982085)
   2) BAI.AT1< 0.1239771 2023  907 Fracaso (0.55165596 0.44834404)
     4) PE.PT1>=0.9406235 207   45 Fracaso (0.78260870 0.21739130)
```

```
       8) lnAC1>=5.266599 147   10 Fracaso (0.93197279 0.06802721) *
       9) lnAC1< 5.266599 60   25 Activa (0.41666667 0.58333333) *
     5) PE.PT1< 0.9406235 1816  862 Fracaso (0.52533040 0.47466960)
      10) T.PC1>=0.07509543 958  388 Fracaso (0.59498956 0.40501044) *
      11) T.PC1< 0.07509543 858  384 Activa (0.44755245 0.55244755) *
   3) BAI.AT1>=0.1239771 433  137 Activa (0.31639723 0.68360277)
     6) CF.PT1< 0.1009116 86   35 Fracaso (0.59302326 0.40697674)
      12) CF.PT1>=-0.1300429 67   16 Fracaso (0.76119403 0.23880597) *
      13) CF.PT1< -0.1300429 19    0 Activa (0.00000000 1.00000000) *
     7) CF.PT1>=0.1009116 347   86 Activa (0.24783862 0.75216138)
      14) lnAT1< 4.740892 13    0 Fracaso (1.00000000 0.00000000) *
      15) lnAT1>=4.740892 334   73 Activa (0.21856287 0.78143713) *

[[17]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1166 Fracaso (0.52524430 0.47475570)
  2) PE.PT1>=-0.009390342 2389 1099 Fracaso (0.53997488 0.46002512)
    4) CF.PT1>=0.2005091 103   13 Fracaso (0.87378641 0.12621359)
      8) CNAE1=1,2,3,5,6,7,8 86    3 Fracaso (0.96511628 0.03488372) *
      9) CNAE1=4,9 17    7 Activa (0.41176471 0.58823529) *
    5) CF.PT1< 0.2005091 2286 1086 Fracaso (0.52493438 0.47506562) *
  3) PE.PT1< -0.009390342 67    0 Activa (0.00000000 1.00000000) *

[[18]]
n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 707 Activa (0.28786645 0.71213355)
   2) BAI.FP1< -0.6299878 26   0 Fracaso (1.00000000 0.00000000) *
   3) BAI.FP1>=-0.6299878 2430 681 Activa (0.28024691 0.71975309)
     6) BAI.FP1>=1.235683 37   8 Fracaso (0.78378378 0.21621622)
      12) BAI.AT1< 0.2379583 26   0 Fracaso (1.00000000 0.00000000) *
      13) BAI.AT1>=0.2379583 11   3 Activa (0.27272727 0.72727273) *
     7) BAI.FP1< 1.235683 2393 652 Activa (0.27246135 0.72753865)
      14) FM.AT1< -0.710585 14   1 Fracaso (0.92857143 0.07142857) *
      15) FM.AT1>=-0.710585 2379 639 Activa (0.26860025 0.73139975) *

[[19]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 806 Activa (0.3281759 0.6718241)
   2) BAI.FP1< -0.6425043 23   0 Fracaso (1.0000000 0.0000000) *
   3) BAI.FP1>=-0.6425043 2433 783 Activa (0.3218249 0.6781751)
     6) IN.AT1>=3.603997 16   0 Fracaso (1.0000000 0.0000000) *
     7) IN.AT1< 3.603997 2417 767 Activa (0.3173355 0.6826645)
      14) BAI.FP1>=1.407173 24   4 Fracaso (0.8333333 0.1666667) *
      15) BAI.FP1< 1.407173 2393 747 Activa (0.3121605 0.6878395) *

[[20]]
n= 2456
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 909 Activa (0.37011401 0.62988599)
   2) V.AC1>=4.012954 165  68 Fracaso (0.58787879 0.41212121)
     4) BAI.AT1< 0.02604451 37   2 Fracaso (0.94594595 0.05405405) *
     5) BAI.AT1>=0.02604451 128  62 Activa (0.48437500 0.51562500)
      10) BAI.AT1>=0.08882178 65  20 Fracaso (0.69230769 0.30769231) *
      11) BAI.AT1< 0.08882178 63  17 Activa (0.26984127 0.73015873) *
   3) V.AC1< 4.012954 2291 812 Activa (0.35443038 0.64556962)
     6) V.AC1>=0.2921464 2201 805 Activa (0.36574284 0.63425716)
      12) V.FP1< 1.592969 251 105 Fracaso (0.58167331 0.41832669) *
      13) V.FP1>=1.592969 1950 659 Activa (0.33794872 0.66205128) *
     7) V.AC1< 0.2921464 90   7 Activa (0.07777778 0.92222222)
      14) BAI.AT1< 0.01447342 13   6 Fracaso (0.53846154 0.46153846) *
      15) BAI.AT1>=0.01447342 77   0 Activa (0.00000000 1.00000000) *

[[21]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1181 Activa (0.4808632 0.5191368)
  2) PE.PT1>=-0.004377676 1451  270 Fracaso (0.8139214 0.1860786)
    4) CF.PT1>=-1.235606 1429  248 Fracaso (0.8264521 0.1735479) *
    5) CF.PT1< -1.235606 22    0 Activa (0.0000000 1.0000000) *
  3) PE.PT1< -0.004377676 1005    0 Activa (0.0000000 1.0000000) *
```

```
[[22]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 1023 Activa (0.4165309 0.5834691)
   2) CF.PT1< 0.5109423 1681  658 Fracaso (0.6085663 0.3914337)
     4) CF.PT1< 0.01960364 554  106 Fracaso (0.8086643 0.1913357) *
     5) CF.PT1>=0.01960364 1127  552 Fracaso (0.5102041 0.4897959)
      10) lnAC1>=7.638869 297   70 Fracaso (0.7643098 0.2356902) *
      11) lnAC1< 7.638869 830  348 Activa (0.4192771 0.5807229) *
   3) CF.PT1>=0.5109423 775    0 Activa (0.0000000 1.0000000) *

[[23]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 2456 1086 Activa (0.4421824 0.5578176)
  2) CF.PT1< 0.4915537 1760  674 Fracaso (0.6170455 0.3829545) *
  3) CF.PT1>=0.4915537 696    0 Activa (0.0000000 1.0000000) *

[[24]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

```
 1) root 2456 751 Activa (0.30578176 0.69421824)
   2) CF.PT1< 0.5112591 2023 751 Activa (0.37123085 0.62876915)
     4) BAI.FP1>=1.364725 71   3 Fracaso (0.95774648 0.04225352) *
     5) BAI.FP1< 1.364725 1952 683 Activa (0.34989754 0.65010246)
      10) BAI.FP1< -0.6413339 56   0 Fracaso (1.00000000 0.00000000) *
      11) BAI.FP1>=-0.6413339 1896 627 Activa (0.33069620 0.66930380) *
   3) CF.PT1>=0.5112591 433   0 Activa (0.00000000 1.00000000) *

[[25]]
n= 2456

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2456 802 Activa (0.326547231 0.673452769)
   2) V.AT1>=0.233671 1945 792 Activa (0.407197943 0.592802057)
     4) PE.PT1>=0.9409179 215  52 Fracaso (0.758139535 0.241860465)
       8) lnAC1>=5.655199 139  11 Fracaso (0.920863309 0.079136691) *
       9) lnAC1< 5.655199 76  35 Activa (0.460526316 0.539473684) *
     5) PE.PT1< 0.9409179 1730 629 Activa (0.363583815 0.636416185)
      10) V.AT1>=3.254659 57   5 Fracaso (0.912280702 0.087719298) *
      11) V.AT1< 3.254659 1673 577 Activa (0.344889420 0.655110580) *
   3) V.AT1< 0.233671 511  10 Activa (0.019569472 0.980430528)
     6) BAI.AT1< -0.01253593 7   0 Fracaso (1.000000000 0.000000000) *
     7) BAI.AT1>=-0.01253593 504   3 Activa (0.005952381 0.994047619) *



sabi.boosting$ponderaciones
  [1] 1.01126965 0.70235931 0.48103035 0.72057257 0.68202438 0.72320805
```

```
 [7] 0.14597931 1.00342619 0.18616019 0.13850101 0.10708498 0.19883357
[13] 0.49022802 0.44299484 0.41668826 0.37027763 0.99953941 0.17354566
[19] 0.17102942 0.00100000 1.03130191 0.66832383 1.01324532 0.15014089
[25] 0.23209252 0.10132251 0.22351919 0.47696672 0.57126357 0.91587758
[31] 0.15514162 0.19544799 0.49537118 0.09392342 0.21839074 0.59032637
[37] 0.11202986 0.16767777 0.91759082 0.16935313 0.30558680 0.01140115
[43] 0.00100000 1.04155564 0.69979010 0.92103010 0.17102942 0.15931478
[49] 0.34169550 0.57794834 0.52357700 0.57571492 0.73116653 0.34078261
[55] 1.04362614 0.18531744 0.17186793 0.19122223 0.16684044 0.08325371
[61] 0.35085496 0.08981729 0.35085496 0.32715367 0.76521221 0.21753711
[67] 0.29668239 0.10379130 0.00100000 1.04570335 0.77501674 1.00537825
[73] 0.18531744 0.25628162 0.08571418 0.20392040 0.69595116 0.43710003
[79] 0.15180696 0.14348478 0.26761120 0.71402112 1.01324532 0.19460229
[85] 0.17690403 0.19883357 0.11780581 0.25454438 0.39280806 0.62494301
[91] 0.12855367 0.00100000 1.01324532 0.77642761 0.41668826 1.02522700
[97] 0.40612692 0.29136168 0.63558985 0.26499081
```

A continuación se recoge la suma ponderada de los votos que han recibido las distintas observaciones para cada clase (fracaso en la primera columna y activa en la segunda), para abreviar se muestran únicamente las veinticinco primeras observaciones de cada clase, en primer lugar las fracasadas y en segundo las activas.

```
>sabi.boosting$votos[1:25,]
         [,1]      [,2]
[1,] 24.04562 18.266272
[2,] 26.84049 15.471399
[3,] 23.83910 18.472793
[4,] 31.35016 10.961729
```

```
 [5,] 27.44445 14.867438
 [6,] 30.78733 11.524561
 [7,] 33.67900  8.632889
 [8,] 26.27640 16.035486
 [9,] 31.98912 10.322773
[10,] 25.88939 16.422502
[11,] 33.31645  8.995442
[12,] 26.11207 16.199824
[13,] 33.76140  8.550493
[14,] 39.07745  3.234438
[15,] 27.38655 14.925340
[16,] 35.03093  7.280959
[17,] 31.45159 10.860304
[18,] 33.03031  9.281581
[19,] 30.95289 11.359002
[20,] 21.71219 20.599701
[21,] 24.84264 17.469249
[22,] 28.01063 14.301262
[23,] 37.71716  4.594733
[24,] 26.65427 15.657618
[25,] 27.77337 14.538521
```

```
 >sabi.boosting$votos[1366:1391,]
          [,1]      [,2]
[1366,] 20.6163904 21.695499
[1367,]  0.8164707 41.495419
[1368,] 22.1868831 20.125006
[1369,] 25.2305743 17.081315
```

```
[1370,]  0.6599542 41.651935
[1371,]  0.0010000 42.310889
[1372,] 21.6968476 20.615042
[1373,]  0.1315537 42.180336
[1374,]  0.3315670 41.980322
[1375,] 19.1985924 23.113297
[1376,]  1.1382189 41.173670
[1377,]  0.0030000 42.308889
[1378,]  0.2148074 42.097082
[1379,]  0.3293872 41.982502
[1380,]  0.1140299 42.197859
[1381,]  0.4977404 41.814149
[1382,]  0.1067913 42.205098
[1383,]  1.3879407 40.923949
[1384,]  0.4351785 41.876711
[1385,] 19.7127747 22.599115
[1386,]  0.1057913 42.206098
[1387,]  1.4178027 40.894087
[1388,]  0.0010000 42.310889
[1389,]  0.0010000 42.310889
[1390,]  0.6564418 41.655448
[1391,]  0.2195371 42.092352

>sabi.boosting$clase
   [1] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
   [8] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
  [15] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
  [22] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
  [29] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
 [36] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [43] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [50] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [57] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [64] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [71] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [78] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [85] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [92] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso"
 [99] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[106] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[113] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[120] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[127] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[134] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[141] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[148] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[155] "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso"
[162] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[169] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[176] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[183] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[190] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[197] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[204] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[211] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[218] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[225] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[232] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
 [239] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [246] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [253] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [260] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [267] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [274] "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [281] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [288] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [295] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [302] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [309] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [316] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [323] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [330] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [337] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [344] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [351] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [358] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [365] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [372] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [379] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [386] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [393] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [400] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [407] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso"
 [414] "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [421] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [428] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [435] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
 [442] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [449] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [456] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [463] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [470] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [477] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [484] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [491] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [498] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [505] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [512] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [519] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [526] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [533] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [540] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Activa"
 [547] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [554] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [561] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [568] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [575] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [582] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [589] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [596] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [603] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [610] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [617] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [624] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [631] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [638] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
[645] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[652] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[659] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[666] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[673] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[680] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[687] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[694] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[701] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[708] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[715] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[722] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[729] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[736] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[743] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[750] "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[757] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[764] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[771] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[778] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[785] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[792] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[799] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[806] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[813] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[820] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[827] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[834] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[841] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
 [848] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [855] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [862] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [869] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [876] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [883] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [890] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [897] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [904] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [911] "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso"
 [918] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [925] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [932] "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [939] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [946] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [953] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [960] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [967] "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso"
 [974] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [981] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [988] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
 [995] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1002] "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso"
[1009] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1016] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1023] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1030] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1037] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1044] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
```

```
[1051] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1058] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1065] "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1072] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1079] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1086] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1093] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1100] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1107] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1114] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1121] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1128] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1135] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1142] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Activa"
[1149] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1156] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Activa"
[1163] "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1170] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1177] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Activa"  "Fracaso" "Fracaso"
[1184] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1191] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1198] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1205] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1212] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1219] "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso" "Fracaso"
[1226] "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1233] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1240] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1247] "Fracaso" "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
```

```
[1254] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso"
[1261] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Fracaso" "Activa"
[1268] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1275] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1282] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1289] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1296] "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1303] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1310] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1317] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1324] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1331] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1338] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1345] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1352] "Activa"  "Fracaso" "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1359] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1366] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Fracaso"
[1373] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1380] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1387] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1394] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1401] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1408] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1415] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1422] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1429] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1436] "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1443] "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1450] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Fracaso"
```

```
[1457] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1464] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1471] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1478] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1485] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1492] "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1499] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1506] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1513] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1520] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1527] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Fracaso"
[1534] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1541] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1548] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1555] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1562] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1569] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1576] "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1583] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1590] "Fracaso" "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"
[1597] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1604] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso"
[1611] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1618] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1625] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1632] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1639] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1646] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1653] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
```

```
[1660] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1667] "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1674] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1681] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1688] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1695] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1702] "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1709] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1716] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"
[1723] "Activa"  "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[1730] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1737] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1744] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[1751] "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Fracaso"
[1758] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1765] "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1772] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1779] "Fracaso" "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[1786] "Activa"  "Fracaso" "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[1793] "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso"
[1800] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1807] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Fracaso" "Activa"
[1814] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso"
[1821] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1828] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1835] "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1842] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1849] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1856] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso"
```

```
[1863] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"
[1870] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"
[1877] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1884] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1891] "Activa"  "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[1898] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1905] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[1912] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1919] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1926] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1933] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso"
[1940] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"
[1947] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1954] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1961] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"
[1968] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[1975] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[1982] "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[1989] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[1996] "Fracaso" "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"
[2003] "Activa"  "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[2010] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2017] "Activa"  "Activa"  "Fracaso" "Activa"  "Fracaso" "Activa"  "Activa"
[2024] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2031] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2038] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2045] "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2052] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2059] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
```

```
[2066] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2073] "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"
[2080] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2087] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2094] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2101] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2108] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2115] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2122] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2129] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[2136] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2143] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2150] "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"  "Activa"
[2157] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2164] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[2171] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2178] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2185] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2192] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2199] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2206] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2213] "Activa"  "Fracaso" "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"
[2220] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2227] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[2234] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[2241] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2248] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2255] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2262] "Activa"  "Fracaso" "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
```

```
[2269] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2276] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2283] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2290] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2297] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2304] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2311] "Activa"  "Fracaso" "Activa"  "Fracaso" "Fracaso" "Activa"  "Activa"
[2318] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2325] "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2332] "Activa"  "Activa"  "Fracaso" "Activa"  "Fracaso" "Fracaso" "Activa"
[2339] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[2346] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2353] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"
[2360] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso"
[2367] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2374] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2381] "Fracaso" "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[2388] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2395] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2402] "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"
[2409] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"
[2416] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2423] "Activa"  "Activa"  "Activa"  "Activa"  "Fracaso" "Fracaso" "Activa"
[2430] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2437] "Activa"  "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"
[2444] "Activa"  "Fracaso" "Activa"  "Activa"  "Activa"  "Activa"  "Activa"
[2451] "Activa"  "Activa"  "Activa"  "Activa"  "Activa"  "Activa"

$importancia
```

```
 JURIDICA   CNAE1    IN.AT1    BAI.AT1   BAI.FP1    AC.AT1    T.AT1

 1.302083  5.729167  3.385417 10.156250 11.197917  4.687500  4.687500

 V.FP1   AC.PC1     T.PC1     FM.V1    PE.PT1    CF.PT1    FM.AT1

5.729167 1.562500  3.645833  1.562500 12.500000 15.104167  2.343750

 V.AT1     V.AC1    lnAT1     lnAC1

7.031250  2.864583 2.604167  3.906250
```